# A First Look at Peer-to-Peer Worms: Threats and Defenses

Lidong Zhou[1], Lintao Zhang[1], Frank McSherry[1], Nicole Immorlica[2,⋆], Manuel Costa[3], and Steve Chien[1]

[1] Microsoft Research Silicon Valley
{lidongz, lintaoz, mcsherry, schien}@microsoft.com
[2] Laboratory for Computer Science, MIT
nickle@theory.lcs.mit.edu
[3] Microsoft Research Cambridge and University of Cambridge
manuelc@microsoft.com

**Abstract.** Peer-to-peer (P2P) worms exploit common vulnerabilities in member hosts of a P2P network and spread topologically in the P2P network, a potentially more effective strategy than random scanning for locating victims. This paper describes the danger posed by P2P worms and initiates the study of possible mitigation mechanisms. In particular, the paper explores the feasibility of a self-defense infrastructure inside a P2P network, outlines the challenges, evaluates how well this defense mechanism contains P2P worms, and reveals correlations between containment and the overlay topology of a P2P network. Our experiments suggest a number of design directions to improve the resilience of P2P networks to worm attacks.

## 1 Introduction

Peer-to-peer (P2P) overlay networks enjoy enormous and ever increasing popularity both in real-life deployment (e.g., Gnutella and KaZaA) and in the research community (e.g., Chord [18], CAN [13], Pastry [14], and Tapestry [24]). While security issues for P2P networks have received attention, the main focus remains on ensuring correct operations within a P2P network in the face of failures and malicious participants. Examples include maintaining the internal structure of a P2P network (e.g., [2]) and fair sharing of resources (e.g., [5]). The threats that a large-scale P2P network deployment poses to Internet security have largely been ignored.

In this paper, we argue that P2P networks provide an ideal venue for new types of worms that prey on common vulnerabilities on the hosts 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 *scanning worms*, which probe addresses randomly for new victims, in three important ways. First, they spread much faster, since they do not waste time probing unused IP addresses. Second, they do not generate high rates of failed connections. Finally, 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

⋆ Work done during internship at Microsoft Research Silicon Valley.

M. Castro and R. van Renesse (Eds.): IPTPS 2005, LNCS 3640, pp. 24–35, 2005.

worms are no longer effective. Because the number of subscribers to a P2P network such as KaZaA is estimated to be in the millions, P2P worms have the potential to compromise a significant fraction of the Internet population. We therefore study the feasibility of constructing a self-defense infrastructure within a P2P network for containing P2P worms. The infrastructure imposes new and challenging requirements for worm-defense mechanisms, while the evaluation of the proposed infrastructure, both analytically and through simulation, reveals interesting correlations between worm containment in a P2P network and the overlay topology of the network. Furthermore, our experiments suggest a number of design directions to improve the resilience of P2P networks to worm attacks.

The rest of the paper is organized as follows. Section 2 elaborates on the imminent threat of P2P worms and makes a case for new defense mechanisms. Section 3 explores possible countermeasures against P2P worms, outlines a self-defense infrastructure, and presents a containment model. The evaluation of the self-defense infrastructure through both theoretical analysis and simulations appears in Section 4. We conclude in Section 5.

## 2 Imminent Threat of P2P Worms

Popular P2P clients such as KaZaA already have a high penetration into the Internet population. Any vulnerability in such a P2P client can put all those hosts at risk. The likelihood of having an exploitable vulnerability in these pieces of software is alarmingly high. A buffer overflow bug in the FastTrack network core, the underlying network for KaZaA and several others, was discovered and disclosed recently [12]. To make things worse, many P2P clients are bundled with spyware, further increasing the chances of introducing intentional or unintentional backdoors into hosts in P2P networks. For example, Saroiu et al. [15] found vulnerabilities in two wide-spread spyware programs due to lack of authentication in their auto-update processes.

Proof-of-concept viruses, such as Gnuman, VBS.Gnutella, and Fizzer [20], which propagate through Gnutella or KaZaA were released in the wild as early as 2000. The impact of these viruses was limited largely because their propagation relied heavily on certain user actions. In contrast, a P2P worm can infect vulnerable hosts automatically by exploiting the same types of vulnerabilities that led to notorious scanning worms such as CodeRed and Slammer. Whereas these random-scanning worms search for new vulnerable hosts by probing "randomly" generated IP addresses, a P2P worm can quickly identify new vulnerable hosts by following the list of neighbors in the overlay topology.

As a form of topological worm [21], P2P worms do not exhibit easily detectable anomalies in network traffic as scanning worms do. A scanning worm has no information on the locations of vulnerable hosts and thus is error-prone in choosing targets; it has to rely on both a reasonable density of vulnerable hosts in the entire IP address space and on the ability to probe different hosts at a high rate. It is these characteristics that lead to schemes for containing scanning worms (e.g., [23,26,7,22]) by detecting and reacting to various network anomalies.

Although these proposed mechanisms show promise for fast detection and successful containment of scanning worms, they have limited power against P2P worms. The

P2P topology provides an accurate way for worms to find more vulnerable hosts without probing random ones; the vastly improved accuracy in identifying vulnerable hosts also eliminates the need to communicate with a large number of different hosts at a high rate. The attack traffic can thus easily blend into normal P2P traffic. Therefore, new defense mechanisms are needed.

## 3 Mitigating Threats of P2P Worms

P2P worms would not exist if we could eliminate vulnerabilities on P2P hosts or cut off a worm's propagation between neighboring P2P hosts. But neither is achievable in practice. To eliminate vulnerabilities, P2P client programs should be written in a type-safe language (e.g., Java or C#), so that it is free of buffer-overflow vulnerabilities. Unfortunately, this is not the case for most existing client programs. Furthermore, common vulnerabilities could exist on co-located software or even the underlying platform. Increased diversity in a P2P network reduces the likelihood of common vulnerabilities and makes it harder for a P2P worm to propagate through P2P neighbors. Further measures can be taken to protect the neighbor list from access by worms. But it is usually hard to distinguish valid accesses from invalid ones.

Given that P2P clients will unlikely be free of common exploitable vulnerabilities in the foreseeable future, an interesting research question is the feasibility of incorporating a self-defense infrastructure into a P2P network for the network itself to detect outbreaks of any unknown worm and contain its spread.

### 3.1 Automatic Detection of Worms

Automatic detection of P2P worms is a prerequisite to any worm containment infrastructure—human responses are simply too slow. Because P2P worms target only hosts in a P2P network, referred to as *nodes*, automatic detection mechanisms must be deployed within the P2P network. We call nodes with automatic worm detection capabilities *guardian nodes*.

Because P2P worms do not exhibit easily detectable anomalies in network behavior, guardian nodes must instead detect worms by identifying the infection process inside running applications. Such detectors can detect broad classes of vulnerabilities. One promising approach, pioneered by several independent research projects [19,4,6,11], is based on the observation that a majority of worms work by hijacking the control flow of a vulnerable program to execute malicious code injected from the network or to force a different execution of code that was already loaded by the program. By tracking how information from untrusted sources propagates its influence in memory during code execution, a worm can be detected when the control flow of the program is arbitrarily controlled by information from untrusted sources. However, the proposed detection mechanisms either require hardware modifications [19,6] or demand expensive binary rewriting/interpretation with significant performance degradation [4,11]. It is therefore reasonable to assume that such general guardian nodes constitute only a small fraction of a P2P population. Since the detection mechanism contains the vulnerable code in a sandboxed environment, we can assume the guardian nodes are invulnerable to worm attacks.

### 3.2 Alert Generation, Propagation, and Processing

With a small fraction of guardian nodes, it is crucial that, once a guardian node detects a worm, it promptly generates a message about the ongoing attack and informs other nodes in the P2P network. We refer to these messages as *alerts*. The purpose of alerts is for a recipient to learn enough information about the attack in order to take appropriate action to become immune to the attack.

Because alerts trigger actions by receiving nodes, an adversary could attack by disseminating bogus alerts. If the receiver of an alert responded by shutting down the vulnerable application, this would turn a worm attack into a denial-of-service attack. To avoid this problem, guardians can generate self-certifying alerts, as described in [4]. Self-certifying alerts are machine-verifiable proofs of vulnerability; they contain a description of the events that lead to a vulnerable behavior—for instance a sequence of network messages—and they can be independently and inexpensively verified by any host. Use of self-certifying alerts also implies that any host can independently decide to become a guardian, since guardians do not have to be trusted. This setting makes it difficult to mount targeted attacks on the guardians. Alternatively, alerts can be submitted to a trusted authority, who verifies the authenticity of the alert and signs the alert using the private key corresponding to a well-known public key. Such an infrastructure for distributing and verifying signed updates already exists in many pieces of software for securing automatic software updates. The trusted authority could be implemented using multiple servers [25] to withstand attacks to a fraction of the servers.

Upon verifying the authenticity of an alert, a host can take several actions to protect itself. For instance, it can stop the vulnerable application or install a new local firewall rule to block a worm "signature"[1]; this could be a simple byte pattern on network messages or a more elaborate signature that accesses network messages and application state. Ideally, a host should identify the vulnerability exploited by the detected attack and patch it automatically. Such patches can be generated locally by the hosts receiving an alert, avoiding the need to trust patches produced by the host that generated the alert. We are currently working towards this goal.

We assume alerts are propagated in the same P2P network as P2P worms. After all, any existing link used by alerts requires that the destination address be recorded on the source; such information is also available to attackers when the source is compromised. This assumption distinguishes our model from that in [4], which also explored the concept of alerts for containment of Internet worms; there, a special P2P network is used for fast and reliable alert dissemination.

### 3.3 A Basic Worm Containment Model

The previous discussions on a self-defense infrastructure yield the following basic model for the containment study. Variations of the basic model are investigated in Section 4.

[1] While several schemes ([16,9,8]) have been proposed for automatic detection of worms and automatic generation of worm signatures, the detection mechanisms rely heavily on the network anomalies that scanning worms exhibit.

Consider a P2P network and a worm that exploits a vulnerability in the nodes of the network. We consider node $A$ a *neighbor* of node $B$ if the address of $A$ appears in node $B$'s state as a P2P client. The topology of a P2P network can be modeled as a directed graph in which each vertex in the graph corresponds to a node in the P2P network and each edge is weighted by the latency of the corresponding link from a node to its neighbor.

Each node in the P2P network has an independent probability $p$ of being a guardian node; otherwise, the node is *vulnerable*. A vulnerable node becomes *infected* when the worm probes this node. A worm starts at a uniformly random node and in each step probes all the neighbors of newly infected nodes. If a worm probes a guardian node, the guardian node will detect the worm, generate an alert, and immediately propagate the alert to its neighbors. A vulnerable node becomes *immune* upon receiving the alert and propagates the alert further to its neighbors. An infected node ignores the alert; it does not propagate it further. Immune nodes do not become infected even upon worm probing. For simplicity, we assume that the worm and the alert incur the same latency on each link, although different links may have different latencies. Furthermore, we ignore the dynamic changes in the P2P network and assume a static topology.

## 4 Analysis and Evaluation

The basic worm containment model characterizes a battle between worm propagation and alert propagation within the same P2P network. The following questions naturally arise.

- With only a small number of guardian nodes, can the self-defense infrastructure contain a P2P worm?
- With a P2P network serving as the battlefield, how can we design and deploy a P2P network to offer an advantage over P2P worms? What strategies can a P2P worm employ to gain advantage?

This section documents our initial efforts to answer these questions. In particular, we evaluate containment of worms as measured by the percentage of vulnerable nodes that are infected when the network reaches a stable state, where neither alerts nor the worm can propagate further. Note that the containment problem is entirely different from the seminal containment study by Moore et al. [10] because that study focused on random probing worms in the Internet.

### 4.1 P2P Network Topology and Worm Containment

**Theoretical analysis.** The topology of a P2P network dictates propagation of a P2P worm and its containment in our basic model. In the absence of guardian nodes, the diameter of the graph, defined to be the longest among the shortest distances between any pair of nodes in the graph, is the upper bound on the amount of time it takes for a worm to compromise the entire P2P network. Here, we show a simple theoretical analysis of worm containment in our basic model.

Suppose a P2P network contains $n$ nodes in a graph of maximum degree $d$, where each node is a guardian node with independent probability $p$. Then for a uniformly random starting infection point, the expected fraction of nodes that become infected is bounded above by $O(n^{\log_d(1-p)})$.

To see this, let $x$ be the starting point of the infection, and consider the shortest path tree from $x$ in the network topology. The key observation is that another node $y$ will become infected if and only if there is no guardian node on the shortest path from $x$ to $y$. Thus the expected number of infected nodes is $\sum_{i=1}^{\ell} n_i(1-p)^i$, where $n_i$ is the number of nodes at depth $i$ in the shortest path tree from $x$. Since the topology has maximum degree $d$, we have that $n_i < d^i$; in fact, it is not hard to see that the worst case occurs when the inequality is tight. A straightforward calculation then yields that the expected fraction of infected nodes in this case is $O(n^{\log_d(1-p)})$.

Although the theoretical analysis offers only a loose upper bound for worm containment in our basic model, it does indicate that the number of nodes in the network, the maximum degree of the graph, and the percentage of guardian nodes are likely the factors influencing the containment result. We use simulations to validate the trends predicated by the theoretical results.

**Simulation setup.** Our experiments were performed on P2P graphs generated using a P2P simulator. Among others, the simulator implements the protocols described in Gnutella 0.4, Gia [3], and Pastry. Nodes in those topologies are randomly placed in a 5050-router Internet topology generated using Georgia Tech's Transit-Stub Internet Topology generator [1] with distance between a pair of nodes computed accordingly.

We further developed an epidemic simulator. This simulator takes as input the P2P topology graph and the probability of a node being a guardian node. For each run, the simulator randomly selects a node in the graph as the initial entry point for the worm and picks a set of guardian nodes according to the specified probability. It then simulates the process of worm propagation and alert propagation (after guardian nodes are triggered.) Each of our experiments takes 500 runs, with different randomly chosen initial infection points and different randomly chosen sets of guardian nodes. We report the mean (over the 500 runs) of the infected fraction, measured as the percentage of infected nodes over the entire vulnerable population. (Note that guardian nodes are excluded from the vulnerable population.)

**Simulation results.** In this set of experiments, we look at Gnutella 0.4 graphs. A Gnutella topology can be modeled as an undirected graph because the neighbor relation is symmetric. (We assume that the weights on links are also symmetric.) When a node joins, it selects a seed node already in the P2P network and performs a random walk to find more nodes as potential neighbors. A node $A$ might refuse to be the neighbor for the joining node if the resulting number of allowed connections for $A$ exceeds the maximum degree allowed. The generated graph is the result of running the P2P simulator for $n$ consecutive joins, where $n$ is the specified number of nodes. No node failures or node leavings are modeled.

We generated a set of Gnutella 0.4 graphs with different settings for minimum/maximum node degrees and total number of nodes. The generated graphs have average degrees that are close to the maximum degrees, indicating that nodes tend to have the same degree. Figure 1 clearly indicates that the infected fraction increases when min/max degrees

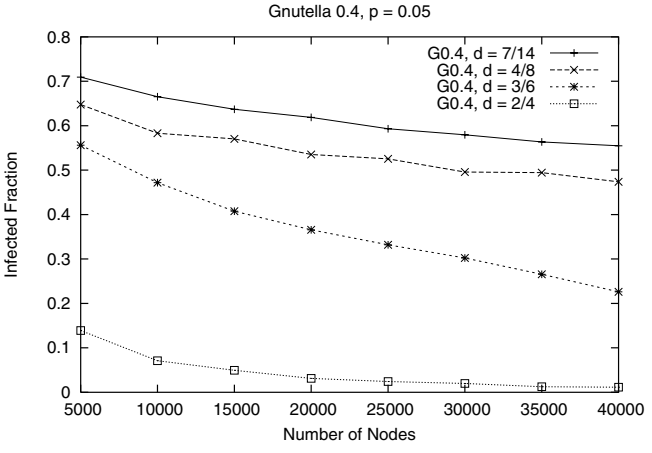


**Fig. 1.** Infected fraction as a function of number of nodes for Gnutella 0.4 graphs with different min/max-degree settings

increase, but decreases when the number of nodes increases, confirming the trends in the theoretical analysis. We want to point out that due to resource limitations, we can only simulate relatively small P2P networks. For real P2P networks with millions of nodes the infection fraction may be significantly lower than the simulation results suggest.

### 4.2 The Effects of Super Nodes

The notion of *super nodes* has been introduced to P2P networks for better scalability. Super nodes are nodes with sufficient resources and high-quality links to accommodate a large number of neighbors. Gia [3] is a proposal to introduce super nodes into Gnutella. In Gia, super nodes emerge as a result of dynamic topology adaptation based on the different capacities of the nodes. Adopting the setting in [3], we set the percentages of nodes at capacity levels 1, 10, 100, 1000, and 10000 at 20%, 45%, 30%, 4.9%, and 0.1%, respectively. Figure 2 shows the infected fraction for a Gia graph, with an average degree around 15 and min/max degrees of 3/128, compared to Gnutella 0.4 graphs with varying min/max degrees. We see a clear downtrend of the infected fraction when the probability of guardian nodes increases and that Gia exhibits the worst containment result.

Super nodes undoubtedly play a significant role in aiding the propagation of the worm due to their high connectivity. It seems that the defense mechanism would be more effective if the choice of guardian nodes were biased towards such high-degree nodes. This is confirmed by the result shown in Figure 3, where, in the case of biased choices of guardian nodes, the probability of a node being a guardian node is proportional to its degree. Note that, even if a worm knows about this strategy and tries to evade detection by biasing against high-degree nodes, the worm propagation will be at a significant disadvantage compared to alert propagation, which is able to exploit the powerful super nodes.

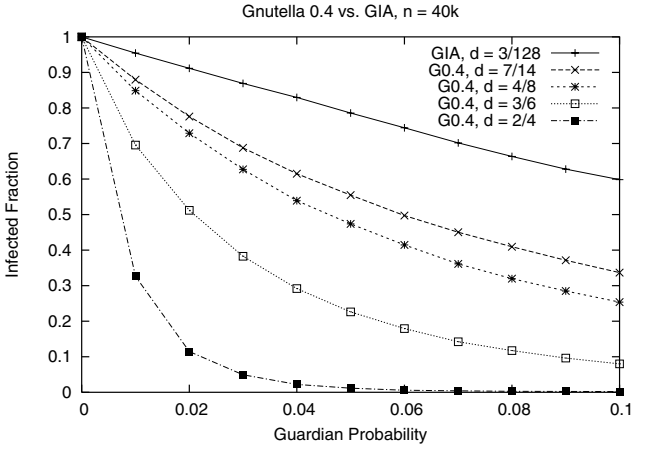


**Fig. 2.** Gnutella 0.4 vs. Gia, 40,000 nodes. Infected fraction as a function of probability of guardian nodes

### 4.3 Hit List and Secret Network of Guardian Nodes

For bootstrapping, P2P networks such as Gnutella and KaZaA offer an initial list of hosts in the network to serve as seed nodes for new nodes to join. An attacker can also collect a large number of addresses through crawling. A P2P worm can use those addresses as an initial hit list [17] instead of starting with a single node.

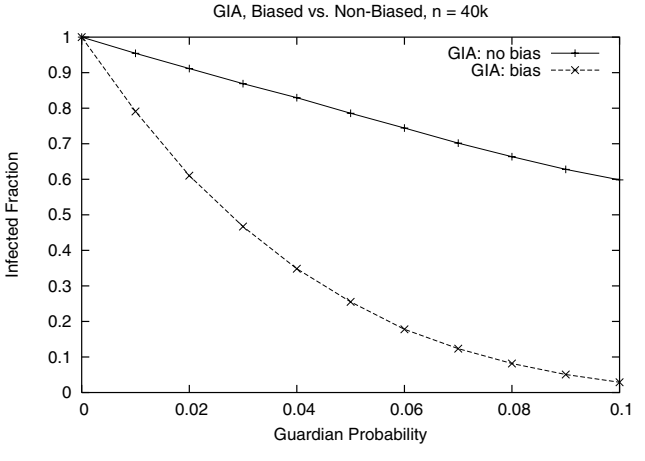


**Fig. 3.** Biased choices of guardian nodes vs. non-biased choices. Gia with 40,000 nodes. Infected fraction as a function of probability of guardian nodes.

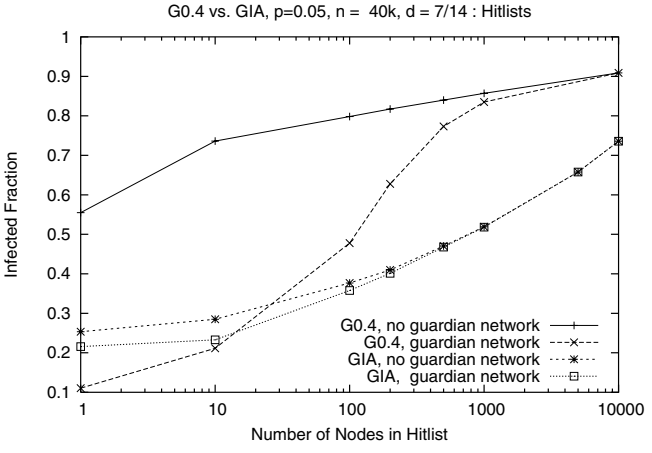


**Fig. 4.** Infected fraction, with and without a secret network of guardian nodes, as the function of number of nodes in the hit list (log scale). Gia with bias and Gnutella 0.4 with min/max degrees of 7/14, 40,000 nodes and 5% of guardian nodes.

In response, guardian nodes could be made aware of each other and form a secret network to tunnel alerts through directly.[2] For simplicity, we assume that this secret network is fully connected with direct links between any two guardian nodes with an average network delay.

Figure 4 shows how the infected fraction reacts to an increasing number of nodes on the hit list, as well as the effects of having a secret network of guardian nodes. Using a hit list seems to be an effective strategy for worms especially when the percentage of the nodes in the hit list becomes significant. Connecting all guardian nodes has a limited advantage in these cases. Define *worm diameter* to be the amount of time for a worm to reach the entire population of the network in the absence of guardian nodes. The effect of connecting the guardian nodes seems to diminish as the worm diameter decreases.

### 4.4 The Effects of Diversity

We have been assuming that the entire population (except for the guardian nodes) is vulnerable. This might not be the case in practice. In particular, P2P clients might use different implementations of the same protocol and run on different hardware/software platforms. Vulnerabilities in one particular implementation or on one particular platform may not affect the entire population due to diversity. The existence of the initially immune nodes works to our advantage because these nodes block worm propagation but pass alerts on to other nodes.

[2] It might seem that we are violating our assumption that the worm and the alerts are propagating in the same topology. This is not the case. In our model, links in the secret network cannot be exploited by worms because guardian nodes are never compromised.

Figure 5 shows the impact of having initially immune nodes in the network. We vary the percentage of the nodes that are initially immune from 0% to 60%. These nodes are chosen uniformly at random. Every node in the set of the non-immune nodes becomes a guardian node with 0.05 probability. The infected fraction shows the percentage of vulnerable nodes (i.e., excluding initially immune nodes and guardian nodes) that are infected. The results show a significant reduction in infected fraction as the immune proportion grows and suggest that diversity is an effective deterrence to P2P worms.

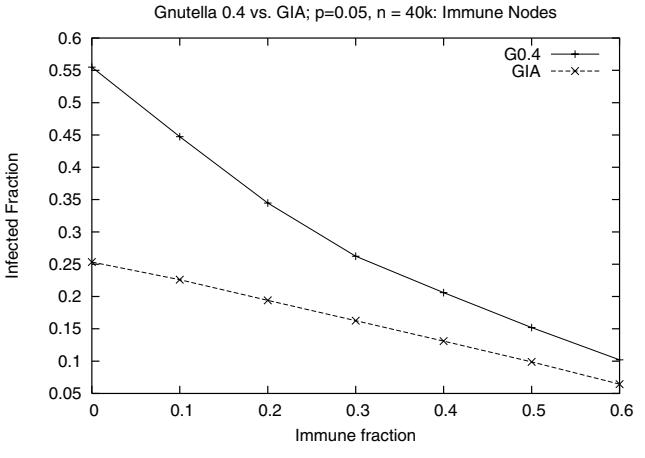


**Fig. 5.** Infected fraction as a function of percentage of immune nodes. Gia and Gnutella 0.4 (max/min degree of 7/14), with 40,000 nodes, 5% guardian nodes, and biased choice of guardian nodes for Gia.

### 4.5 Design Implications for P2P Networks

In summary, our experiments suggest a number of design directions over which P2P networks could evolve to increase their resilience to worm attacks. First, P2P protocols should bias their choice of neighbors to maximize diversity. Second, mechanisms should be included to make crawling the overlay more difficult or impossible. Otherwise, an attacker can gain a substantial advantage by building a large initial hit list to launch the worm. Finally, mechanisms should exist to deploy guardian nodes at flexible locations in the P2P network. As our preliminary results show, placement of these nodes has an important effect on containment.

## 5 Concluding Remarks

P2P worms constitute a potentially deadly threat to Internet security, a threat that we are not yet prepared for. This paper outlines a self-defense infrastructure to be built into a P2P network for containing P2P worms. The proposed infrastructure not only poses

new challenges to worm-containment research, but also gives rise to an interesting phenomenon of competing epidemics (worm vs. worm-triggered alerts) in a P2P network.

The paper represents our initial study on containment of P2P worms with debatable assumptions. We plan to explore further the feasibility of the self-defense infrastructure, investigate more topologies and new strategies, and work towards a unifying theory that identifies the defining characteristics of the network topology on worm containment. Such a theory would help predict worm containment for a given topology and help develop strategies to improve defense against P2P worms, because applying those strategies can always translate into some network topology transformation.

## Acknowledgements

The authors would like to thank Martín Abadi, Úlfar Erlingsson, Chandu Thekkath, and Ted Wobber, as well as the anonymous reviewers, for their helpful suggestions.

## References

1. K. Calvert, M. Doar, and E. Zegura. Modeling Internet topology. *IEEE Communications Magazine*, June 1997.
2. M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and S. S. Wallach. Secure routing for structured peer-to-peer overlay networks. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI '02)*, pages 299–314, Boston, MA, USA, December 2002. USENIX.
3. Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making Gnutella-like p2p systems scalable. In *Proceedings of SIGCOMM'03*, pages 407–418, Karlsruhe, Germany, August 2003. ACM.
4. M. Costa, J. Crowcroft, M. Castro, and A. Rowstron. Can we contain Internet worms? In *Proceedings of the 3rd Workshop on Hot Topics in Networks (HotNets-III)*, November 2004.
5. L. P. Cox and B. D. Noble. Honor among thieves in peer-to-peer storage. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, pages 120–132, Bolton Landing, NY, USA, November 2003. ACM SIGOPS, ACM Press.
6. J. R. Crandall and F. T. Chong. Minos: Control data attack prevention orthogonal to memory model. In *Proceedings of the 37th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE/ACM, December 2004.
7. J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan. Fast portscan detection using sequential hypothesis testing. In *Proc. 25th Symposium on Security and Privacy*. IEEE, May 2004.
8. H. Kim and B. Karp. Autograph: Toward automated, distributed worm signature detection. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.
9. C. Kreibich and J. Crowcroft. Honeycomb—creating intrusion detection signatures using Honeypots. In *Proc. of the 2nd Workshop on Hot Topics in Networks (HotNets-II)*, November 2003.
10. D. Moore, C. Shannon, G. Voelker, and S. Savage. Internet quarantine: Requirements for containing self-propagating code. In *Proceedings of IEEE INFOCOM 2003*. IEEE, March 2003.
11. J. Newsome and D. Song. Dynamic taint analysis: Automatic detection and generation of software exploit attacks. In *Proceedings of the 12th Annual Network and Distributed System Security Symposium (NDSS 2005)*, Feb 2005. To Appear.

12. random nut. The PACKET 0' DEATH FastTrack network vulnerability. NETSYS.COM Full Disclosure Mailing List Archives, May 2003. http://www.netsys.com/full-disclosure/2003/05/msg00351.html.
13. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proceedings of ACM SIGCOMM*, pages 161–172, San Diego, CA, USA, August 2001.
14. A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proc. IFIP/ACM Middleware 2001*, Heidelberg, Germany, Nov. 2001.
15. S. Saroiu, S. D. Gribble, and H. M. Levy. Measurement and analysis of spyware in a university environment. In *Proceedings of the 1st Symposium on Networked Systems Design and Implementation (NSDI)*, San Francisco, CA, March 2004.
16. S. Singh, C. Estan, G. Varghese, and S. Savage. The EarlyBird system for real-time detection of unknown worms. Technical Report CS2003-0761, UC San Diego, August 2003.
17. S. Staniford, V. Paxson, and N. Weaver. How to 0wn the Internet in your spare time. In *Proceedings of the 11th USENIX Security Symposium*, August 2002.
18. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proc. ACM SIGCOMM*, pages 149–160, 2001.
19. G. E. Suh, J. Lee, and S. Devadas. Secure program execution via dynamic information flow tracking. In *Proceedings of ASPLOS XI*, pages 85–96, Boston, MA, USA, October 2004.
20. http://securityresponse.symantec.com/.
21. N. Weaver, V. Paxson, S. Staniford, and R. Cunningham. A taxonomy of computer worms. In *The First ACM Workshop on Rapid Malcode (WORM)*, 2003.
22. N. Weaver, S. Staniford, and V. Paxson. Very fast containment of scanning worms. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.
23. M. M. Williamson. Throttling viruses: Restricting propagation to defeat malicious mobile code. In *Proc. 18th Annual Computer Security Applications Conference*, Las Vegas, NV, Dec. 2002.
24. B. Y. Zhao, L. Huang, S. C. Rhea, J. Stribling, A. D. Joseph, and J. D. Kubiatowicz. Tapestry: A global-scale overlay for rapid service deployment. *IEEE Journal on Selected Areas in Communications (J-SAC)*, 22(1):41–53, January 2004.
25. L. Zhou, F. B. Schneider, and R. van Renesse. COCA: A secure distributed on-line certification authority. *ACM Transactions on Computer Systems*, 20(4):329–368, November 2002.
26. C. Zou, L. Gao, W. Gong, and D. Towsley. Monitoring and early warning for Internet worms. In *Proc. of the 10th ACM Conference on Computer and Communication Security*, Oct. 2003.