# On the Privacy of Peer-Assisted Distribution of Security Patches

Di Wu[§], Cong Tang[‡], Prithula Dhungel[†], Nitesh Saxena[†], Keith W. Ross[†]
[§]Sun Yat-Sen University, Guangzhou, China
[‡]Peking University, Beijing, China
[†]Polytechnic Institute of NYU, Brooklyn, USA

***Abstract*—When a host discovers that it has a software vulnerability that is susceptible to an attack, the host needs to obtain and install a patch. Because centralized distribution of patches may not scale well, peer-to-peer (P2P) approaches have recently been suggested. There is, however, a serious privacy problem with peer-assisted patch distribution: when a peer $A$ requests a patch from another peer $B$, it announces to $B$ its vulnerability, which $B$ can exploit instead of providing the patch. Through analytical modeling and simulation, we show that a large majority of vulnerable hosts will typically become compromised with a basic design for peer-assisted patch distribution. We then study the effectiveness of two different approaches in countering this privacy problem. The first approach utilizes special-purpose peer nodes, referred to as *honeypots*, that discover and blacklist malicious peers listening for patch requests from susceptible hosts. In the second approach, the patches are requested through an *anonymizing network*, hiding the identities of susceptible hosts from malicious peers. Using analytical models and simulation, we show that, honeypots do not completely solve the privacy problem; in contrast, an anonymizing network turns out to be more suitable for security patch distribution.**

## I. Introduction

The current decade has seen rapid growth in large-scale cyber attacks, such as viruses, worms, trojans and other types of malware. Some prominent examples include the ILoveYou virus, Code Red Worm, Storm Worm and most recently Conficker. These attacks have increased in frequency as well as strength, infecting millions of computers and causing enormous financial losses world-wide. A natural defense strategy to counter these attacks is to distribute updated software, called *patches*, to vulnerable hosts. Timely distribution of security patches is a challenging problem due to the large number of hosts that need to be patched as well as the speed at which the attacks spread. A patch distribution method should be able to patch a large fraction of susceptible hosts before they become compromised. The traditional approach for patch distribution is centralized in nature, whereby patches are distributed through one or more of software vendor's servers or through a Content Distribution Network (CDN). Such a centralized patching approach has been adopted by automated patch distribution tools (e.g., Microsoft Windows Update, Symantec Update, McAfee VirusScan).

Centralized patch distribution can be costly for vendors. Moreover, it may lead to severe server-side bottlenecks (e.g., patch traffic generated by Windows Update is huge [1]), thus undermining the speed of patching as well as its scalability. To remedy the problems with centralized patch distribution, there has been a growing interest among the research community to design Peer-to-Peer (P2P) based solutions. With P2P patch distribution, susceptible hosts can request patches from previously patched hosts, rather than relying (solely) on a centralized infrastructure. A number of P2P-flavored patch distribution methods have been suggested recently [2], [3], [4], [5]. (We will review these methods in the following section.)

A peer-assisted patch distribution system typically includes a *"tracker"* (which may be centralized or decentralized with a Distributed Hash Table) that tracks the peers that currently have the patch. A peer $A$ that wants the patch, first contacts the tracker, which instructs $A$ to obtain the patch from one of the patched peers, say, peer $B$. $A$ then contacts $B$ and downloads the patch. Once patched, $A$ can register with the tracker and serve as a patch provider for other unpatched peers. Recent experimental results show that peer-assisted patch distribution can be used to effectively meet security patch distribution deadlines and minimize server bandwidth [5].

There is, however, a serious privacy problem with peer-assisted patch distribution: when a peer $A$ requests a patch from another peer $B$, it announces to $B$ its vulnerability, which $B$ can exploit instead of providing the patch. Thus, an attacker can obtain the patch and register with the tracker, wait for patch requests from vulnerable peers, and then exploit their vulnerability right after. We remark that the privacy problem is intrinsic to all pull-based P2P systems. As an example, in file-sharing and video streaming P2P systems, a peer would reveal what song, movie or program it is interested in. However, in the context of P2P patch distribution, the privacy problem is clearly much more critical and potentially devastating.

**Our Contributions:** We make the following two-fold contributions in this paper:

- We develop a fluid model to bring insight to the aforementioned privacy problem in peer-assisted patch distribution. In particular, we show that typically a large majority of vulnerable hosts will become compromised with a basic design for peer-assisted patch distribution.
- We study the effectiveness of two different approaches in countering the privacy problem. The first approach utilizes special-purpose peer nodes, referred to as *honeypots*, that discover and blacklist malicious peers listening for patch requests from susceptible hosts. In the second

approach, the patches are requested through an *anonymizing network*, hiding the identities of susceptible hosts from malicious hosts. To study both approaches, we again develop fluid models, which provide insights into when these approaches successfully patch the majority of susceptible hosts before they are compromised by attacking peers. We conclude that the two approaches – honeypots and anonymization – can improve the effectiveness of peer-assisted patch distribution. However, honeypots do not completely solve the privacy problem, particularly when the attackers scan the system with a very fast rate. On the contrary, an anonymizing network turns out to be more suitable for delivering security patches, primarily due to their small sizes [5] and low-bandwidth requirements. This is unlike other high-bandwidth applications that typically suffer from performance issues when run over an anonymizing network.

We note that honeypots and anonymizing networks are well-known concepts in network security and privacy. However, this paper is the first (to the best of authors' knowledge) to ascertain, via theoretical and simulation-based evaluation, their effectiveness towards solving the privacy problem in peer-assisted distribution of security patches.

**Organization:** In Section II, we review prior work related to modeling propagation of cyber-attacks (such as worms) and patch distribution approaches. In Section III, we demonstrate the impact that the privacy problem could cause on peer-assisted patch distribution. In Sections IV and V, we study the applicability of honeypots and anonymizing network to address the privacy problem.

## II. Related Work

To reduce the cost of centralized patch distribution, peer-assisted patch distribution has attracted significant attention in recent years. In [2], Shakkottai et al. proposed a pull-based P2P mechanism for patch distribution. In [1], Gkantsidis et al. closely examined the Microsoft Windows Update system and demonstrated that P2P is promising for large-scale patch dissemination. In [6], [7], Xie et al. proposed a patching scheme based on connected dominating set for unstructured P2P file-sharing systems. In [5], Serenyi et al. implemented a BitTorrent-style patch distribution system that can guarantee timely patch delivery by using central planning.

In the past years, researchers have also conducted theoretical studies on the propagation and containment of worms and viruses. Most of research work (e.g., [8], [9], [10], [11], [12]) focused on the modeling of the natural spread of malware in different types of networks (e.g., the Internet and unstructured P2P networks). The classic epidemiology model is a commonly used theoretical tool. Shakkottai et al. explicitly derived the time taken to effectively combat the progress of the worm, and the maximum number of infected hosts under both server-based and P2P-based approaches [2]. Vojnovic et al. studied the race between worm propagation and patch dissemination, and quantified the speed of patch or alert dissemination required for worm containment [3], [4].

However, none of these research efforts take the critical privacy problem into account. To the best of our knowledge, this is the first paper to address the privacy issues in peer-assisted patch distribution. In addition to developing an analytical theory for modeling and analysis, we also investigate the effectiveness of two different approaches – honeypots and anonymization – to counter the privacy problem.

## III. Basic Model for Peer-Assisted Patch Distribution

### A. Basic Design for Peer-Assisted Patch Distribution

In peer-assisted patch distribution, a client who needs to acquire the security patch, first requests the patch from a infrastructural patch server. After receiving the request, if the patch server is not too busy, it uploads the patch to the client; otherwise, it redirects the client to a peer that possesses the patch. After having downloaded the patch successfully, the client then registers itself with the server (or with a centralized tracker or decentralized DHT tracker), since it can now upload the patch to other peers. During the registration process, the client may have to present to the server a proof that it indeed possesses the patch in consideration (e.g., by presenting a hash of the patch).

One major problem with this basic P2P approach is that attackers can easily discover vulnerable peers. Specifically, an attacker can join the system and request the patch from the server or another peer, if the server is busy. After having downloaded the patch, it registers itself as a patch distribution peer. A susceptible peer may then be re-directed to the attacking peer for obtaining the patch. In such an event, the attacker, instead of uploading the patch to the vulnerable client, exploits the vulnerability at the client to compromise the client machine. We refer to this problem as the *privacy problem in peer-assisted patch distribution*.

### B. Modeling and Analysis of P2P Patch Distribution

In this section, we describe our basic fluid model for P2P patch distribution, which is amenable for analysis and provides insights into the privacy problem of patch distribution.

We define a *susceptible peer* as any uncompromised peer without the patch. Susceptible peers seek to acquire and install the patch. As shown in Figure 1, a susceptible peer issues patch request to the *tracker*, which redirects the client to either the *patch server* or one of the peers in the P2P distribution network. Note here that the responsibilities of the tracker and the patch server have been distributed to two different hosts. A *benevolent peer* is defined as any uncompromised peer with the patch[1]; benevolent peers register with the tracker so that they can assist with the patch distribution. We define an *attacking peer* as any peer that seeks to compromise susceptible peers; once an attacking peer obtains a patch from the patch server or another peer, it can register with the tracker.

In developing our fluid model, we make the following reasonable assumptions: $(i)$ the bottleneck is upload capacity

[1]We assume that once patched, a peer cannot get compromised and conversely, once a peer is compromised, it cannot be patched.

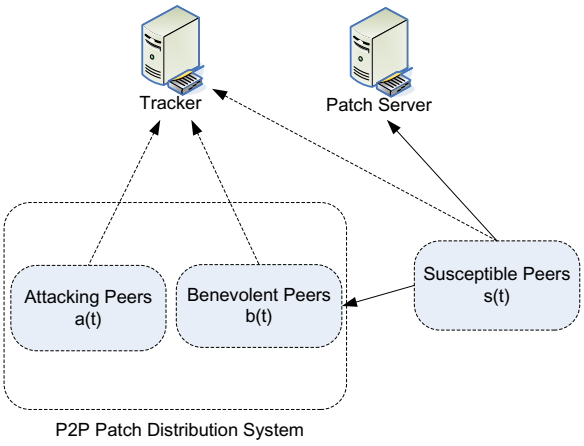



Fig. 1. P2P Patch Distribution

and not the download capacity; $(ii)$ peers have homogeneous upload bandwidth; $(iii)$ susceptible peers first request the patch from the server; if the server is busy, then they will request patches from the peers. Without loss of generality, the patch size is normalized to be 1. We will use the following notations in our fluid model.

- $u_s$ is the upload rate of server, in units of patch size;
- $u_p$ is the upload rate of peers, in units of patch size;
- $b(t)$ is the number of benevolent peers at time $t$;
- $a(t)$ is the number of attacking peers at time $t$;
- $s(t)$ is the number of susceptible peers at time $t$;
- $n$ is the total number of peers; thus $b(t)+a(t)+s(t) = n$;
- $\lambda$ is the rate at which a susceptible peer requests patches from the patch distribution system.
- $p(t)$ is the probability that a peer requests a patch from a benevolent peer, when the server is busy; that is, $p(t) = \frac{b(t)}{b(t)+a(t)}$.

Recall that our primary goal is to study the impact of the privacy problem in P2P patch distribution. To this end, in the model presented in this section, we assume that peers are only compromised when they expose themselves to an attacking peer. We do not consider the possibility of peers being directly compromised via active scanning or propagation of the vulnerability in question. In the subsequent sections, however, we will also allow attacking peers to perform active scanning to find susceptible peers.

We now describe the evolution of the number of benevolent peers, $b(t)$, the number of attacking peers, $a(t)$, and the number of susceptible peers, $s(t)$. Note that $a(t)$ and $b(t)$ are increasing functions, while $s(t)$ is a decreasing function. Also note that $s(t)\lambda$ is the rate of patch requests sent by all the susceptible peers. We assume throughout that $s(0)\lambda > u_s$, i.e., initially the patch server does not have sufficient capacity to meet the demand for patches. (Otherwise, there would be no need for peer-assisted patch distribution.) Let

$$t_2 = \min\{t \geq 0 : s(t)\lambda \leq u_s\} \tag{1}$$

We identify two phases in the evolution of the susceptible peers. In the *Initial Phase*, when $0 \leq t \leq t_2$, the patch server does not have enough upload bandwidth to handle all patch requests. In this case, the server can only serve a fraction of patch requests. The unsatisfied patch requests $(s(t)\lambda - u_s)$ will be directed towards peers in the patch distribution system. For a given unsatisfied patch request, a requesting peer will either be served by benevolent peers with probability $p(t)$, or be compromised by attacking peers with probability $1 - p(t)$. However, not all the requests can be satisfied by benevolent peers as the service rate of benevolent peers is also constrained by their total upload capacity $b(t)u_p$. Thus, in the Initial Phase, the number of benevolent peers increases at a rate of $u_s + \min\{(s(t)\lambda - u_s) \cdot p(t), b(t)u_p\}$. Meanwhile, as the patch requests are directed to attacking peers at a rate of $(s(t)\lambda - u_s) \cdot (1 - p(t))$, attacking peers can compromise all the requesters and thus their number increases at the same rate. Thus, the evolution of $a(t)$ and $b(t)$ in the *Initial Phase* are governed by the following fluid equations:

$$\begin{cases} b'(t) = u_s + \min\{(s(t)\lambda - u_s) \cdot p(t), b(t)u_p\} \\ a'(t) = (s(t)\lambda - u_s) \cdot (1 - p(t)) \\ b(t) + a(t) + s(t) = n \\ b(0) = m_1, a(0) = m_2 \end{cases} \tag{2}$$

Here, $m_1$ and $m_2$ are the initial number of benevolent peers and attacking peers, respectively.

After the end of the Initial Phase at $t = t_2$, the system enters the *Final Phase* during which $s(t)\lambda \leq u_s$. During this final phase, the number of attacking peers no longer grows; thus, $a(t) = a(t_2)$ for all $t \geq t_2$.

The Initial Phase can be further decomposed into two phases. To see this, let

$$t_1 = \min\{t \geq 0 : (s(t)\lambda - u_s) \cdot p(t) \leq b(t)u_p\}$$

*Lemma 1:* $t_1 < t_2$.

*Proof:* Because $s(t)$ is a monotonically decreasing function, $t_1 < t_2$ is true iff $s(t_1) > s(t_2)$. When $(s(0)\lambda - u_s) \cdot p(t) > b(t)u_p$, we have $t_1 > 0$ and $s(t_1) = \frac{u_s + nu_p}{\lambda + u_p}$; otherwise, $t_1 = 0$ and $s(t_1) = n - m_1 - m_2$. In both cases, if $n - m_1 - m_2 > u_s/\lambda$, $s(t_1) > s(t_2) = u_s/\lambda$ always holds. As $s(0) = n - m_1 - m_2$, we can derive that when $s(0) > u_s/\lambda$, $t_1 < t_2$ holds. ■

We refer to the period $0 \leq t \leq t_1$ as the *Early Initial Phase* and the period $t_1 \leq t \leq t_2$ as the *Late Initial Phase*. In the *Early Initial Phase*, the benevolent peers do not have enough upload bandwidth to satisfy all the incoming patch requests. Note that if $t_1$ equals zero, the Early Initial Phase does not occur. By solving the differential equations, we have the following proposition:

*Proposition 1:* In the Early Initial Phase when $0 \leq t \leq t_1$, the evolution of benevolent peers $b(t)$ is given by

$$b(t) = (m_1 + \frac{u_s}{u_p}) \cdot e^{u_p t} - \frac{u_s}{u_p}$$

In the Late Initial Phase when $t_1 \le t \le t_2$, the evolution of $b(t)$ and $a(t)$ can be described by the following closed-form expressions:

$$\begin{cases} b(t) = n - c_1 e^{-\lambda(t-t_1)} - c_2 \frac{n - c_1 e^{-\lambda(t-t_1)}}{(e^{\lambda(t-t_1)} - \frac{c_1}{n})^{\frac{u_s}{n\lambda}}} \\ a(t) = c_2 \frac{n - c_1 e^{-\lambda(t-t_1)}}{(e^{\lambda(t-t_1)} - \frac{c_1}{n})^{\frac{u_s}{n\lambda}}} \end{cases}$$

where

$$\begin{cases} c_1 = n - b(t_1) - a(t_1) \\ c_2 = \frac{a(t_1)}{b(t_1)+a(t_1)} (\frac{b(t_1)+a(t_1)}{n})^{u_s/n\lambda} \end{cases}$$

In the Final Phase when $t \ge t_2$, the evolution of $b(t)$ and $a(t)$ is given by

$$\begin{cases} b(t) = n - a(t_2) - [n - b(t_2) - a(t_2)] e^{-\lambda(t-t_2)} \\ a(t) = a(t_2) \end{cases}$$

*Proof:* In the Early Initial Phase, we can simplify Equation (2) as:

$$\begin{cases} b'(t) = u_s + b(t) u_p \\ a'(t) = (s(t)\lambda - u_s) \cdot (1 - p(t)) \\ b(t) + a(t) + s(t) = n \\ b(0) = m_1, a(0) = m_2 \end{cases} \quad (3)$$

We can derive $b(t) = (m_1 + \frac{u_s}{u_p}) \cdot e^{u_p t} - \frac{u_s}{u_p}$. As the equation of $a(t)$ is non-linear, $a(t)$ does not have explicit solution and can only be solved numerically.

In the Late Initial Phase, we can simplify the equations as follows:

$$\begin{cases} b'(t) = u_s + (s(t)\lambda - u_s) \cdot p(t) \\ a'(t) = (s(t)\lambda - u_s) \cdot (1 - p(t)) \\ b(t) + a(t) + s(t) = n \end{cases} \quad (4)$$

In the beginning of this phase, the number of benevolent and attacking peers are given by $b(t_1)$ and $a(t_1)$. We can obtain the following closed-form solutions:

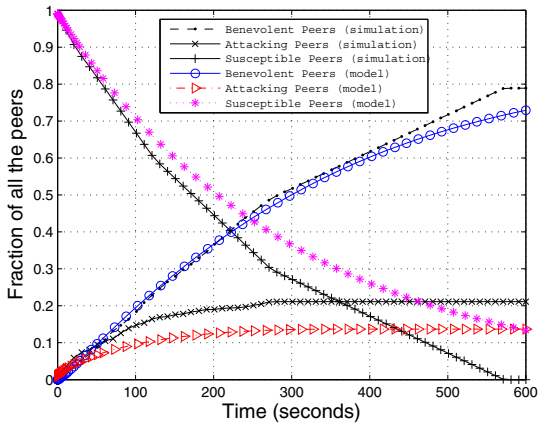
$$\begin{cases} b(t) = n - c_1 e^{-\lambda(t-t_1)} - c_2 \frac{n - c_1 e^{-\lambda(t-t_1)}}{[e^{\lambda(t-t_1)} - \frac{c_1}{n}]^{\frac{u_s}{n\lambda}}} \\ a(t) = c_2 \frac{n - c_1 e^{-\lambda(t-t_1)}}{[e^{\lambda(t-t_1)} - \frac{c_1}{n}]^{\frac{u_s}{n\lambda}}} \end{cases}$$

where $c_1 = n - b(t_1) - a(t_1)$ and $c_2 = \frac{a(t_1)}{b(t_1)+a(t_1)} (\frac{b(t_1)+a(t_1)}{n})^{u_s/n\lambda}$. In the Final Phase, which is the easiest to analyze, the evolution of the benevolent peers is simply governed by $b'(t) = \lambda s(t) = \lambda[n - b(t) - a(t)]$. After solving this equation we have $a(t) = a(t_2)$ and $b(t) = n - a(t_2) - [n - b(t_2) - a(t_2)] e^{-\lambda(t-t_2)}$.

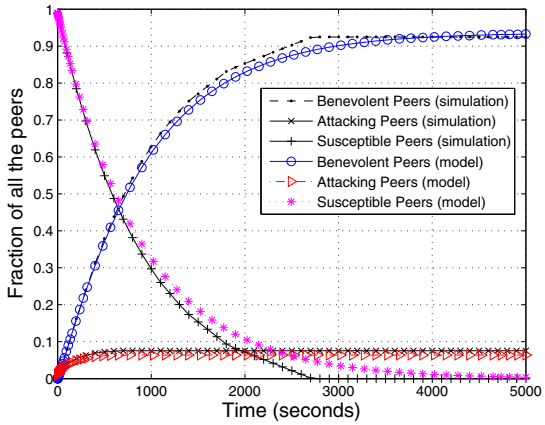
■

### C. Model Validation

We developed a discrete event simulator to validate our fluid model, for both small ($n = 1,000$) and large systems ($n = 10,000$). In the default settings, the patch size is set as 125KB, which is within the range of typical patch sizes reported by [5]. The peer upload bandwidth is set to 200Kbps. The server bandwidth, in the small and large system, is set as 1Mbps and 5Mbps, respectively. In the simulation, both the server and benevolent peers maintain a fixed-size queue and serve the patch requests based on FIFO policy. When the queue is full, the patch request is dropped. The discrete-event simulation is more realistic than the fluid model, as perfect scheduling of patch distribution is assumed in the fluid model. For each system, we run the simulation five times and take the average over all the runs.




(a) Small System $n = 1,000, m_1 = 10, m_2 = 0, \lambda = 1/300$




(b) Large System $n = 10,000, m_1 = 100, m_2 = 0, \lambda = 1/900$

Fig. 2. Validation of Fluid Model: our fluid model conforms well with the simulation results.

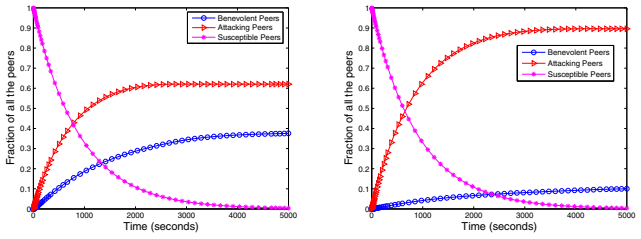
Figure 2 compares the simulation results with the analytical results obtained from Proposition 1. For the small system, we observe that the fluid model generally captures the evolution trend of benevolent and attacking peers. For the large system, the fluid model provides an excellent approximation of simulation curves. In the real world, as there exists a large number of susceptible peers, fluid model is well-suited for our theoretical analysis.

In the following sections (**including Sec IV and Sec V**), we also validate, via extensive simulation, all our results obtained through the fluid model. As our simulation results conform well with the analytical results, for clarity of presentation, we only show analytical results henceforth.

### D. Impact of System Parameters

In this section, we study the impact of system parameters on the effectiveness of peer-assisted patch distribution.

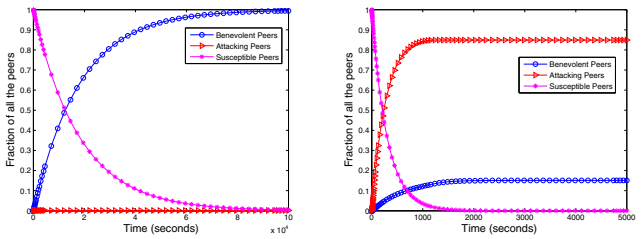
In the default setting, the patch size is set as 125KB. The server and peer bandwidth is set as 5Mbps and 200Kbps, respectively. The peer requests the patch every 15 minutes (i.e., $\lambda = 1/900$). Similar to the configuration in [4], [5], the total number of peers in the system $n$ is set to 100,000. The initial numbers of benevolent peers and attacking peers are set as 0 and 100 respectively. The configuration of the initial number of benevolent/attacking peers represents the case that attacking peers can register with the tracker in the early stage of patch distribution.




(a) $u_s = 5$ Mbps (b) $u_s = 1$ Mbps

Fig. 3. Impact of server upload bandwidth: when the server bandwidth is high, the long-run fraction of attacking peers decreases greatly.

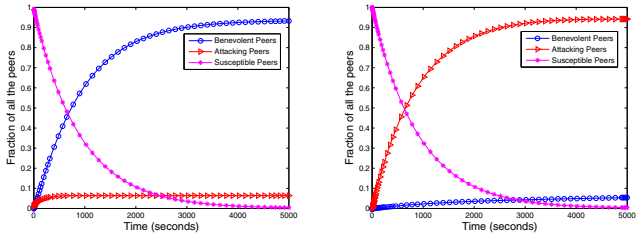
*1) Impact of server upload bandwidth:* Figure 3 shows the impact of server upload bandwidth $u_s$ on P2P patch distribution. It can be seen that the rate of increase of attacking peers is more for lower values of $u_s$. For lower values of server bandwidth, more patch requests will be directed to the peers and so the probability of attacking peers being selected as the patch providers is increased, thus increasing the rate of increase of attacking peers. With an increase of the server bandwidth by a factor of 5, the long run fraction of attacking peers decreases by 30%.




(a) every 5 hours, $\lambda = 1/18000$ (b) every 5 minutes, $\lambda = 1/300$

Fig. 4. Impact of patch request rate (a) Every 5 hours: when the request rate is very low, all the requests can be handled by the server and there is no increase in the number of attacking peers. (b) Every 5 minutes: when increasing the request rate, the long-run fraction of attacking peers also increases.

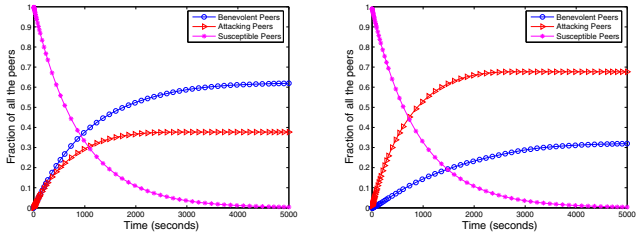
*2) Impact of patch request rate:* Figure 4 shows the impact of patch request rate. For larger values of request rate $\lambda$, the total number of patch requests per unit time is greater and so larger fraction of patch requests are forwarded to the peers. This increases the probability that attackers get selected as patch providers, therefore increasing the rate of increase of attacking peers. We notice that, when the patch request rate is increased by a factor of 60, the long run fraction of attacking peers increases by 85%.




(a) $n = 10,000$ (b) $n = 1,000,000$

Fig. 5. Impact of network size: when the network is larger, a larger fraction of susceptible peers will be compromised.

*3) Impact of network size:* Figure 5 shows the evolution of benevolent and attacking peers in the system for different values of network size. For a larger network size, the value of $s(t)\lambda$ is larger. That is, the number of patch requests per unit time is greater and so a larger fraction of patch requests are forwarded to the peers. Similar to previous cases, this increases the rate of increase of attacking peers. Increasing the network size by a factor of 100 increases the long run fraction of attacking peers by 90%.




(a) $m_1 = 100, m_2 = 100$ (b) $m_1 = 0, m_2 = 1000$

Fig. 6. Impact of the initial number of benevolent peers and attacking peers (for $n = 100,000$): $m_1$ and $m_2$ are the initial number of benevolent peers and attacking peers respectively.

*4) Impact of the initial numbers of benevolent peers and attacking peers:* Figure 6 shows the impact of the initial number of benevolent peers and attacking peers on the system evolution. Initially, if there are a large number of attacking peers in the system, it is with higher probability that an attacking peer will be selected as the patch distributor when the patch server is busy. On the contrary, in case that there are a large number of initial benevolent peers, the attack becomes less successful.

To summarize, our results of this section show that even while not considering the active propagation of the vulnerability via scanning, malicious patch peers can have an adverse effect and the privacy problem can indeed have serious consequences. We observed that the system ends up being in a state where there is a large majority of malicious peers. A few exceptions to this are restrictive scenarios where (1) the server

upload bandwidth is very high; (2) patches are requested at a slow rate (e.g., every 5 hours); (3) the network size is small; (4) there exist a large number of benevolent or patched peers at the system initialization.

## IV. Defending Against Attackers by Using Honeypots

We have just observed that if attacking peers can register with the tracker during the early stage of the patch distribution, they can compromise a large fraction of susceptible peers. We now study the effectiveness of a *honeypot-based approach* in reducing the fraction of compromised peers with peer-assisted patch distribution. A honeypot is a specialized machine that is intentionally set up to detect and blacklist attackers who attempt to infiltrate unauthorized computer systems. Honeypots have been widely used in intrusion detection systems [13], identifying email spammers [14], infiltrating botnets [15], etc. In our design, the purpose of customized honeypots is to identify and blacklist attacking peers.

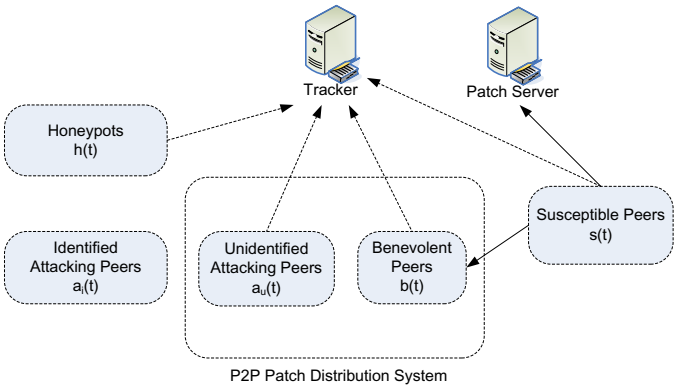



Fig. 7. P2P patch distribution with honeypots

To achieve the above goal, a honeypot periodically retrieves peer lists from the tracker and sends patch requests to the peers in the list at random. Note that since the tracker and honeypot nodes are under the control of the vendor, the tracker server can be configured to send non-intersecting sets of IP addresses to different honeypot machines. This improves the efficiency by having the honeypot system cover a wider range of IP addresses per unit time. After acquiring a list of peers, the honeypot will issue a patch request to each peer on the list. Some of these peers will be benevolent peers and the remainder will be malicious; the honeypot does not know a priori which ones are malicious. If the honeypot issues a request to an attacking peer, the attacking peer will likely try to compromise the honeypot. On the other hand, if the honeypot issues a request to a benevolent peer, the benevolent peer will upload the patch. By observing the pattern of incoming byte stream, the honeypot will be able to identify whether the remote peer is malicious or not. To improve efficiency, a honeypot can launch multiple connections to different peers simultaneously. After detecting an attacking peer, the honeypot can ask the tracker to blacklist the IP address corresponding to the attacking peer, that is, not direct a patch requesting peer to this IP address. The architecture of P2P patch distribution system augmented with honeypots is shown in Figure 7.

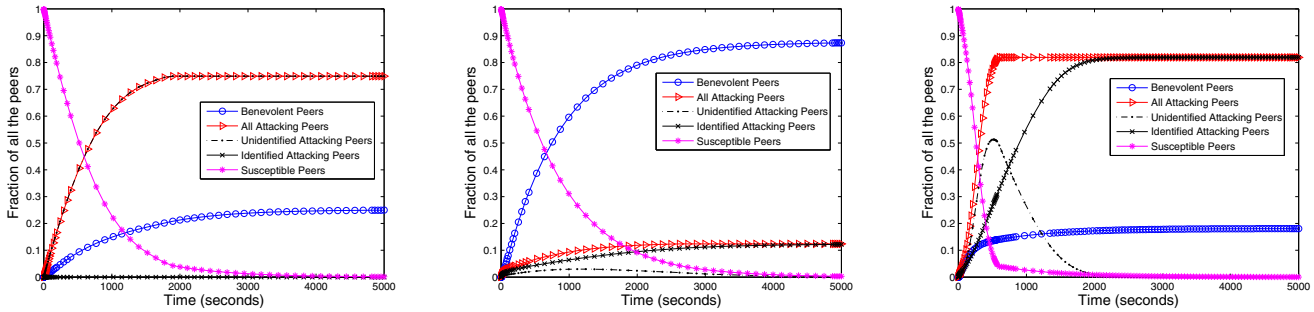
Before modeling the effects of honeypots, we make the following reasonable assumptions: $(i)$ honepots are well protected machines and can themselves not be attacked; $(ii)$ honeypots are maintained by the vendor and that the attackers do not know or cannot easily determine the list of honeypot nodes; $(iii)$ maintaining a honeypot requires a considerable amount of attention and resources, so that there is a limit on the number of honeypots that a vendor can potentially deploy.

Let $h$ be the number of honeypots deployed and $\lambda_h$ be the rate at which a honeypot sends out patch requests. Define attacking peers before and after being identified by the honeypots as "*unidentified attacking peers*" and "*identified attacking peers*," respectively. We use $a_u(t)$ and $a_i(t)$ to denote the number of unidentified and identified attacking peers, respectively. If the patch server is busy, the probability that a susceptible peer (or honeypot) requests the patch from a benevolent peer is given by $p(t) = \frac{b(t)}{b(t)+a_u(t)}$. The definition of $p(t)$ is similar to that in the previous section. The only difference is that $a(t)$ (the total number of attacking peers) is replaced by $a_u(t)$ (the number of unidentified attacking peers) in the expression.

In addition to passively listening for patch requests from susceptible peers, we now assume that attacking peers also perform active scanning to compromise peers with the underlying vulnerability. (We did not consider this in the previous section, since the attackers were highly successful even without active scanning.) Let $\beta$ denote the rate at which per-host scans hit a peer in the patch distribution system. The evolution of benevolent peers, unidentified attacking peers and identified attacking peers can be depicted by Equation (5) ( $(\cdot)$ is an indicator function).

$$\begin{cases} b'(t) = \quad (s(t)\lambda < u_s) \cdot s(t)\lambda + \quad (s(t)\lambda \geq u_s) \cdot \\ \qquad [u_s + \min\{(s(t)\lambda - u_s) \cdot p(t), b(t)u_p\}] \\ a'_u(t) = \quad (s(t)\lambda \geq u_s) \cdot (s(t)\lambda - u_s) \cdot (1 - p(t)) \\ \qquad -h\lambda_h(1 - p(t)) + \beta[a_u(t) + a_i(t)] \cdot \frac{s(t)}{n} \\ a'_i(t) = h\lambda_h(1 - p(t)) \\ b(t) + a_u(t) + a_i(t) + s(t) = n \\ p(t) = \frac{b(t)}{b(t)+a_u(t)} \\ b(0) = m_1, a_u(0) = m_2, a_i(0) = 0 \end{cases} \tag{5}$$

In the second expression of this equation, since honeypots actively remove attacking peers from the patch distribution system, the number of unidentified attacking peers $a_u(t)$ decreases at the rate of $h\lambda_h(1-p(t))$. The number of identified attacking peers $a_i(t)$ increases at the same rate. Meanwhile, due to the effects of active scanning, $a_u(t)$ will also increase at the rate of $\beta[a_u(t) + a_i(t)] \cdot \frac{s(t)}{n}$.




(a) No honeypots are used, scanning rate $\beta = 0.001$ (b) 80 honeypots are used, scanning rate $\beta = 0.001$ (c) 80 honeypots are used, scanning rate $\beta = 0.01$

Fig. 8. Impact of honeypots: the use of honeypots can decrease the rate at which nodes are compromised, but it is less effective when attackers perform aggressive scanning.

### A. *Effectiveness of Honeypots*

Figure 8 shows the impact of the number of deployed honeypots and the scanning rate on the effectiveness of honeypot-augmented P2P patch distribution approach. We observe that deploying honeypot nodes indeed decreases the rate at which nodes are compromised. By increasing the number of honeypot nodes from 0 to 80 (as shown in Figure 8(a) and Figure 8(b)), the long run fraction of compromised peers in the network reduces by 60%. However, by comparing Figure 8(b) and Figure 8(c), we observe that, even by using honeypots, when increasing the scanning rate by a factor of 10, the long run fraction of attacking peers increases by 70%. Thus, honeypots do not completely solve the privacy problem, particularly when the attackers scan the system with a very fast rate. In fact, due to the high cost involved in deploying a large number of honeypots, the honeypot approach may not be suitable for large-scale patch distribution.

## V. Patch Acquisition via an Anonymizing Network

In this section, we describe our design and modeling of a P2P patch acquisition mechanism based on a low-latency anonymizing network.

### A. *Anonymizing Network*

The purpose of an anonymizing network is to hide the identity (e.g., IP address) of the sender of a message from its recipient as well as any attacker eavesdropping over the communication. An anonymizing network is typically based upon onion routing [16] and involves routing one's messages (and thereafter receiving responses) over a routing path consisting of a series of (e.g., 3) machines (called *relay nodes*) distributed all over the internet. The message is sent in multiple (e.g., 3) encrypted layers such that the first relay node (called entry node) on the path learns the source of the message but not the message itself or the identity of intended recipient, the last relay node (called exit node) learns the message[2] and its destination but not the source, each intermediary relay node only learns the identity of the relay node from which it received the packet and the identity of the relay node to which the (decrypted) packet is to be forwarded to. However, in case *all* relay nodes on the routing path are malicious and collude with one another, the sender can not remain anonymous. When a number of nodes route messages over an anonymizing network, these different messages get "mixed" and it becomes quite hard for an attacker to break the anonymity of the system. To further confuse the attacker, nodes can themselves be inserting cover traffic into the system.[3] However, in practice, low-latency anonymizing network (ones which do not inject any cover traffic), can be used to provide a reasonable level of anonymity. Tor [21] is an example of low-latency anonymizing network used on the Internet today. Although Tor allows clients/senders to also act as relays, currently it consists of 1000 or so relay nodes, which can be discovered using a centralized directory service. More scalable and purely P2P-based designs of low-latency anonymizing networks have been proposed in the literature [22], [23], [24]. Among these, [24] is particularly interesting for large-scale anonymous communication as it employs a purely decentralized lookup protocol to locate relay nodes.

### B. *Our Design and Model*

An anonymizing network is a natural and straightforward approach to solve the privacy problem with P2P patch distribution. Although patches are delivered over a series of multiple hops, our intuition was that this will only be marginally slower than direct patch delivery since security patches tend to be small in sizes (40–150KB) [5]. However, a careful design choice in building an anonymizing patch distribution system is necessary. One might think that a low-latency P2P anonymous network, such as the one proposed in [24], can be applied in a straight-forward manner. However, an anonymizing network, solely used for the patch distribution service, may not be better than direct patch acquisition. This is because an attacker receiving a request from a susceptible peer (e.g., as an entry

[2]To prevent the exit node from learning the message, the message can be encrypted with destination's public key, for example.

[3]We note that low-latency anonymizing networks have been shown to be vulnerable to a class of traffic analysis attacks, e.g., [17], [18], [19], [20]. However, these attacks often require the attacker to perform long-term and persistent traffic analysis, and might not be practical always. Thus, we do not consider such attacks in our design and modeling.

node) can simply deduce that the request is indeed for a patch and in turn exploit the vulnerability rather than processing the request. To this end, we propose to embed a P2P anonymizing patch distribution network (e.g., the one that uses the system of [24]) into a general-purpose anonymizing network, such as Tor. The resulting system will be used for a large number of privacy-oriented services and applications (such as anonymous web surfing, anonymous chat), including patch acquisition, thus making it harder for the attackers to distinguish patching requests from requests corresponding to other applications. Figure 9 depicts the architecture of the proposed anonymizing patch distribution system.

In the system, we set-forth the following requirements:

- Every peer who has previously acquired a patch (including an attacker) joins the anonymizing network and serves as a relay node.[4] These relay nodes will be in addition to the existing Tor relays.
- When a susceptible peer requests a patch, it randomly selects a peer possessing the patch as the exit node.[5] (The list of patched peers is obtained from the tracker or more scalably, this can be achieved through the P2P lookup method of [24]). This is done because if a susceptible (i.e., unpatched) peer serves as an exit node, it can itself get compromised by attacking peers. The requester can select other relays (excluding the exit node) on the routing path from the entire anonymizing network (including Tor relays).

Having described our anonymizing P2P patch distributions system, we are now ready to develop an analytical model for the same. Let us denote as $r$ the number of relay nodes on the routing path (normally $r = 3$). Let $\rho$ be the bandwidth utilization factor of benevolent peers. When there are multiple benevolent peers on the routing path, the patch will be uploaded multiple times from one hop to the other. In case that all the $r$ relays on the routing patch are benevolent peers, $\rho = 1/(r+1)$. In case only the exit node and the patch supplier are benevolent, which are necessary conditions for a susceptible peer to get a patch, $\rho = 1/2$. In the ideal case, while not considering the traffic of other applications, the range of $\rho$ is given by $\frac{1}{r+1} \leq \rho \leq \frac{1}{2}$. Let $n_a$ denote the number of relays (excluding attacking peers and benevolent peers) in the anonymizing network.

The state transitions in our model are as follows:

- *Susceptible-to-Benevolent:* A patch requester can get a genuine patch only if $(1)$ the patch supplier is benevolent, and $(2)$ the exit node is benevolent. If the exit node is an attacker, it may drop the patch request. The probability of occurrence is given by:

$$q_{sb} = P(\text{supplier is benevolent}) \cdot P(\text{exit node is benevolent})$$

[4]We assume that the tracker employs a method that allows it to validate whether a peer possesses the patch, and if so, registers this peer.

[5]A peer possessing a patch might either be an attacking peer or a benevolent peer.

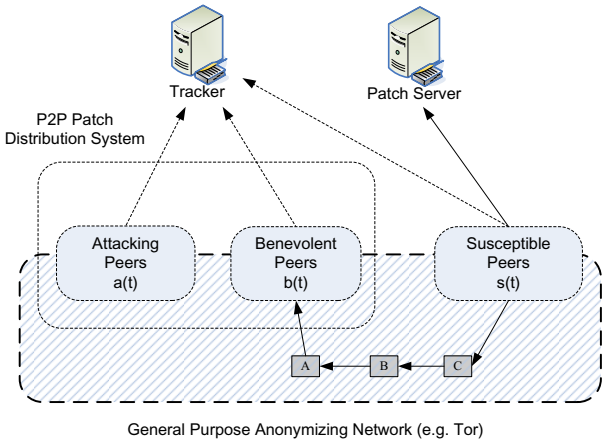



Fig. 9. Anonymizing P2P Patch Distribution System

- *Susceptible-to-Compromised:* A patch requester can be compromised only if the routing path is composed of all attackers. The probability of occurrence is given by:

$$q_{sc} = P(\text{all the relay nodes in the path are malicious})$$

- *Susceptible-to-Susceptible:* In all other situations, the patch requester will not be compromised, however, it would also not be able to obtain the patch.

The fraction of attacking peers in the anonymizing network is given by

$$\theta_a(t) = a(t)/(n_a + b(t) + a(t)) \tag{6}$$

The fraction of benevolent peers is given by

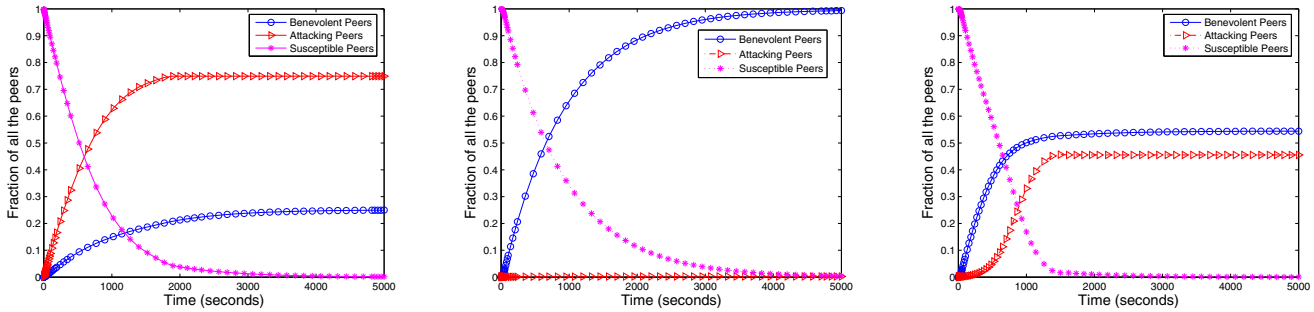
$$\theta_b(t) = b(t)/(n_a + b(t) + a(t)) \tag{7}$$

Thus, we can derive the probabilities $q_{sb}$, $q_{sc}$ as:

$$\begin{cases} q_{sb}(t) = p(t)^2 = (\frac{b(t)}{b(t)+a(t)})^2 \\ q_{sc}(t) = \theta_a^{r-1}(t) \cdot (1 - p(t)) \\ \qquad = (\frac{a(t)}{(n_a+b(t)+a(t))})^{r-1} \cdot \frac{a(t)}{b(t)+a(t)} \end{cases} \tag{8}$$

Similar to our model presented in Section IV, we also consider active scanning performed by the attacking peers. As before, we denote $\beta$ as the rate at which per-host scan by an attacking peer hits a peer in our system.

To calculate the value of $\rho$, let us consider the scenario wherein the requester can successfully download a patch. In this scenario, both the patch supplier and exit node are assumed to be benevolent peers. Let $X$ be the number of other benevolent peers (excluding the exit node and supplier) on the routing path, $0 \leq X \leq r-1$. When $X = k$, the bandwidth utilization factor $\rho = \frac{1}{k+2}$. The average bandwidth utilization factor at time $t$ is:




(a) Anonymizing network is not used, $\beta = 0.001$ (b) Anonymizing network is used, $r = 3, n_a = 1000, \beta = 0.001$ (c) Anonymizing network is used, $r = 3, n_a = 1000, \beta = 0.01$

Fig. 10. Impact of Anonymizing Network on Patch Delivery ($r$ is the number of relay nodes on the routing path, $n_a$ is the number of relays in the anonymizing network, and $\beta$ is the active scanning rate)

$$\begin{aligned}\overline{\rho(t)} &= \sum_{k=0}^{r-1} \frac{1}{k+2} P(X=k) \\ &= \sum_{k=0}^{r-1} \frac{1}{k+2} \binom{r-1}{k} {\theta_b}^k \cdot (1-\theta_b)^{r-1-k} \end{aligned} \quad (9)$$

The evolution of benevolent, susceptible and attacking peers in our model are depicted as follows.

$$\begin{cases} b'(t) = \ (s(t)\lambda < u_s) \cdot s(t)\lambda + \ (s(t)\lambda \geq u_s) \cdot \\ \qquad [u_s + \min\{(s(t)\lambda - u_s) \cdot q_{sb}(t), b(t)u_p \cdot \overline{\rho(t)}\}] \\ a'(t) = \ (s(t)\lambda \geq u_s) \cdot (s(t)\lambda - u_s) \cdot q_{sc} + \beta a(t) \cdot \frac{s(t)}{n} \\ b(t) + a(t) + s(t) = n \\ p(t) = \frac{b(t)}{b(t)+a(t)} \\ b(0) = m_1, a(0) = m_2 \end{cases} \quad (10)$$

### C. Effectiveness of Anonymizing Network

Figure 10 shows the impact of anonymizing network on patch delivery (the new parameters used to obtain these results are: $n_a = 1000$ and $r = 3$; other parameters were kept the same as in our basic model). The plot depicts the evolution of the fraction of benevolent and attacking peers, for two active scanning rates $\beta = 0.001$ and $\beta = 0.01$. Comparing Figure 10(b) with Figure 10(a) we can see the gain in terms of long-run fraction of benevolent peers while using the anonymizing network (note that all the other parameters for Figure 10(a) and Figure 10(b) are the same); when using an anonymizing network, the long-run fraction of benevolent peers is increased by 75%. This can be credited to two reasons: (1) since the probability of all relay nodes on the routing path being malicious is very low, attackers have less chance of compromising susceptible peers when compared to the naive P2P distribution, and (2) although the patches are delivered via multiple hops when using the anonymizing network, reasonably small sizes of security patches slows down patch acquisition only marginally compared to direct patch delivery.

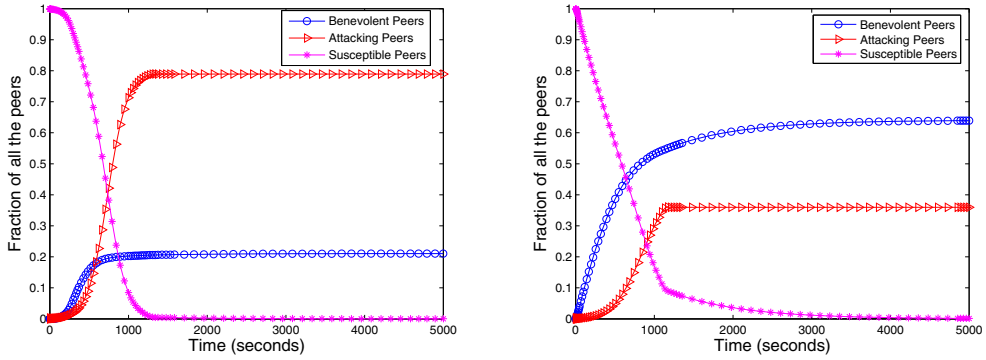
Figure 10(b) and Figure 10(c) depict the impact of the scanning rate on the evolution of benevolent and attacking peers in an anonymizing network. It can be noted that increasing the scanning rate by a factor of 10 increases the long run fraction of attacking peers by 45%. This is an expected result because higher the scanning rate, larger is the fraction of compromised peers. A larger fraction of compromised peers in turn implies a higher probability that all attacking peers belong to the routing paths selected by a susceptible peers, whenever the patch server is busy. This will eventually lead to a higher long-run fraction of attacking peers in the system.

Next, we evaluate the impact of patch size on the effectiveness of the anonymizing patch distribution system. The graphs shown in Figure 10(b) and Figure 10(c), which are for patches of size 125KB, clearly show the usefulness of anonymizing network for patch delivery. As expected, for smaller patch sizes, the anonymizing network is even more effective, as shown in Figure 11(b) for a patch size of 67.5KB. On the other hand, as patches become larger, the anomymizing network fails to curb attacking peers, as shown in Figure 11(a) for a patch of size 625KB. However, as we discussed earlier, since security patches are typically small (40KB – 150 KB), one can conclude that security patch distribution over an anonymizing network can be quite effective.

Finally, comparing Figure 10(b) and Figure 8(b), we notice that anonymizing network is more effective than the use of honeypots in controlling attacking peers. Comparing Figure 10(c) and Figure 8(c) (under a higher active scanning rate) also reveals the same fact.

## VI. Conclusion

In this paper, we focused on the privacy of peer-assisted distribution of security patches. We developed a fluid model to study the impact that malicious peers can have on the effectiveness of patch distribution. Our results show that (even while not considering the active propagation of the vulnerability via scanning) malicious peers can have an adverse effect. We observed that the system ends up being in a state where there is a large majority (70–90%) of malicious peers.




(a) Patch size = 625KB, $r = 3, n_a = 1000, \beta = 0.01$ (b) Patch size = 67.5KB, $r = 3, n_a = 1000, \beta = 0.01$

Fig. 11. Impact of patch size: for smaller patch sizes, the anonymizing network is more effective.

To counter the privacy problem, we considered two different approaches: honeypots and anonymizing network. We again developed fluid models for these approaches in order to investigate their effectiveness in curbing the flow of compromised peers in the system. In these models, we incorporated active propagation of the vulnerability, in addition to considering the attackers who simply listen for patch requests.

Our results show that for an active scanning rate ($\beta$) of 0.001, the fraction of compromised hosts reduces from 75% (i.e., when no honeypots are deployed) down to little over 10%, when using 80 honeypots. However, with an active scanning rate of 0.01, even 80 honeypots were not found to be effective, leading to as high as 80% compromised hosts in the system. The use of anonymizing network, on the other hand, proves quite effective in patching almost all susceptible peers over the long-run, when the vulnerability spreads at the rate of 0.001. Even with a 10 times faster scan rate, a touch above 50% of the susceptible peers were found to be successfully patched, as opposed to less than 10% if the approach were not used. We also found that the anonymizing network service is only suitable for acquiring security patches which tend to be small in sizes. Based on our results, we can conclude that both honeypots and anonymizing network can improve the effectiveness of peer-assisted patch distribution, with the latter being more promising.

## ACKNOWLEDGEMENTS

This work has been in part supported by NFS grant CNS-0917767, the Fundamental Research Funds for the Central Universities, Sun Yat-Sen University (grant no. 2009-35000-3161425/09LGPY56) and Sun Yat-Sen University "Hundred Talents Program" (grant no. 35000-3226138).

## REFERENCES

[1] C. Gkantsidis, T. Karagiannis, and M. Vojnovic, "Planet Scale Software Updates," in *SIGCOMM*, 2006, pp. 423–434.
[2] S. Shakkottai and R. Srikant, "Peer to Peer Networks for Defense Against Internet Worms," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 9, pp. 1745–1752, 2007.
[3] M. Vojnovic and A. J.-. Ganesh, "On the effectiveness of automatic patching," in *WORM*, 2005, pp. 41–50.
[4] M. Vojnovic and A. J. Ganesh, "On the race of worms, alerts, and patches," *IEEE/ACM Trans. Netw.*, vol. 16, no. 5, pp. 1066–1079, 2008.
[5] D. Serenyi and B. Witten, "RapidUpdate: Peer-Assisted Distribution of Security Content," in *IPTPS*, 2008.
[6] L. Xie and S. Zhu, "A Feasibility Study on Defending Against Ultra-Fast Topological Worms," in *Peer-to-Peer Computing*, 2007, pp. 61–70.
[7] L. Xie, H. Song, and S. Zhu, "On the Effectiveness of Internal Patching Against File-Sharing Worms," in *ACNS*, 2008, pp. 1–20.
[8] D. Moore, C. Shannon, and K. C. Claffy, "Code-red: a case study on the spread and victims of an internet worm," in *Internet Measurement Workshop*, 2002, pp. 273–284.
[9] S. Staniford, V. Paxson, and N. Weaver, "How to Own the Internet in Your Spare Time," in *USENIX Security Symposium*, 2002, pp. 149–167.
[10] I. Hamadeh, J. Hart, G. Kesidis, and V. Pothamsetty, "A preliminary simulation of the effect of scanning worm activity on multicast," in *PADS*, 2005, pp. 191–198.
[11] G. Kesidis, I. Hamadeh, Y. Jin, S. Jiwasurat, and M. Vojnovic, "A model of the spread of randomly scanning internet worms that saturate access links," *ACM Trans. Model. Comput. Simul.*, vol. 18, no. 2, 2008.
[12] K. K. Ramachandran and B. Sikdar, "Modeling malware propagation in gnutella type peer-to-peer networks," in *IPDPS*, 2006.
[13] P. Kabiri and A. A. Ghorbani, "Research on intrusion detection and response: A survey," *International Journal of Network Security*, vol. 1(2), no. 84-102, 2005.
[14] Website for the project honeypot, http://www.projecthoneypot.org/.
[15] T. H. et al., "Measurements and Mitigation of Peer-to-Peer-based Botnets: A Case Study on Storm Worm," in *USENIX LEET*, 2008.
[16] D. Chaum, "Untraceable Electronic Mail, Return Addresses, and Digitial Pseudonyms," in *1981*.
[17] X. Wang, S. Chen, and S. Jajodia, "Network Flow Watermarking Attack on Low-Latency Anonymous Communication Systems," in *IEEE Symposium on Security and Privacy*, 2007.
[18] V. Shmatikov and M.-H. Wang, "Timing Analysis in Low-Latency Mix Networks: Attacks and Defenses," in *European Symposium on Research in Computer Security (ESORICS), year = 2006,*.
[19] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker, "Low-resource routing attacks against tor," in *ACM workshop on Privacy in electronic society (WPES)*, 2007.
[20] N. Hopper, E. Y. Vasserman, and E. Chan-Tin, "How much anonymity does network latency leak?" in *ACM Conference on Computer and Communications Security*, 2007.
[21] Tor Anonymizing Network, http://www.torproject.org/.
[22] M. J. Freedman and R. Morris, "Tarzan: a peer-to-peer anonymizing network layer," in *ACM conference on Computer and communications security (CCS)*, 2002.
[23] M. Rennhard and B. Plattner, "Introducing MorphMix: peer-to-peer based anonymous Internet usage with collusion detection," in *ACM workshop on Privacy in the Electronic Society (WPES)*, 2002.
[24] A. Nambiar and M. Wright, "Salsa: a structured approach to large-scale anonymity," in *ACM conference on Computer and communications security (CCS)*, 2006.