A Reputation-based System for the Quarantine of Random Scanning Worms

SCOTT E. COULL
coulls@cs.rpi.edu

BOLESLAW K. SZYMANSKI
szymansk@cs.rpi.edu

Rensselaer Polytechnic Institute
Troy, NY

ABSTRACT

The Internet infrastructure, which has become so critical in our everyday lives, needs automated protection from worm attacks. We propose a system that uses the ideas of reputation and recommendation to effectively quarantine random scanning worms on the Internet. A trust model, based on the passing of recommendations among participating autonomous systems, is used to modulate a localized reputation value that indicates the notoriety, or disrepute, of the attacker. This allows us to enact localized blocking of worm traffic commensurate with the notoriety of the attacker, eventually leading to a global blocking strategy for globally notorious attackers. Unlike previous attempts at collaborative worm containment, our approach's reliance on a trust model provides resilience to Byzantine failures and allows it to be used within the Internet at-large, rather than restricting it to enterprise networks within a single administrative domain. Using simulations, we demonstrate that our reputation-based system is capable of quickly quarantining worms that are many orders of magnitude more virulent than worms released into the Internet thus far. We also discuss the possible uses of the system in quarantining other types of malicious behavior, such as spam and viruses, which have wide-reaching global effects.

1. INTRODUCTION

One of the key shortcomings of today's Internet architecture is the implicit trust placed in hosts. This implicit trust means that there are no consequences built into the architecture of the Internet for malicious or abusive behavior. The consequences that may exist are strictly enacted by human administrators of the various networks and systems connected to the Internet. However, with the ever increasing occurrence of automated attacks, especially random scanning worms, human mitigated responses are simply too slow to effectively protect the Internet at-large from these abuses [10]. There are, of course, best practices that service providers can follow to prevent such attacks, such as ingress filtering to prevent the spoofing of IP addresses from their customer autonomous systems, and 'black holing' abusive prefixes to drop their malicious data as it enters the autonomous system. Unfortunately, the continued occurrence of random scanning worms in the Internet suggests that these measures are either sporadically adopted, or wholly ineffective in a localized setting. What is needed is a collaborative, autonomic, and Internet-wide system for quarantining this malicious behavior quickly, and fairly. We propose a system based on the same principles that humans use to determine good behavior from bad behavior – recommendations and reputation. The general idea of this quarantining system is to utilize existing intrusion and worm detection technologies available in the autonomous systems and stub networks of the Internet to detect worm traffic and issue recommendations to near-by autonomous systems about the malicious behavior of the attacking host. The recommendations of near-by autonomous systems, also referred to as nodes in this paper, are then used to block traffic from offending hosts without having to directly witness a worm attack. Significant corroboration of the malicious behavior must be received from several sources before blocking the traffic to ensure the fair use of the system. This blocking would then have the effect of cutting off Internet usage to the host, or hosts, that are causing the attacks, thereby forcing the user or administrator of those hosts to check their computers for signs of worm infection, while preventing the continued spread of the worm to other vulnerable hosts.

The novelty of our system stems from the fact that we borrow heavily from our daily interactions as humans, and use that experience to create a trust model based on reputation that has not yet been attempted in worm containment research. To make educated decisions, humans typically seek out the advice of others whom they trust the most, namely the friends and family with whom they have close associations. They then use this advice, along with their own experiences, to make intelligent and informed decisions about situations with which they have no direct experience. We can use such a trust model to decide when a host is performing malicious acts without having to directly witness these acts. Of course, because this system is meant to be implemented in the un-trusted Internet, security is of utmost concern. Therefore, we leverage the trust provided in the Border Gateway Protocol (BGP) peering relationships as a basis for our trust model – if a neighboring autonomous system can be trusted to properly route your data then it can also be trusted to provide reliable recommendations about worm traffic. This is a good base, but of course it would be foolish to create a Byzantine situation in which only one node can provoke another node to block any host at will, therefore we require a level of corroboration from multiple sources. This trust model allows us to provide an extremely flexible and responsive system for quarantining random scanning worm traffic while allowing participating nodes to make autonomous decisions based on the recommendations of others. Other systems designed to quarantine worms suffer from possible Byzantine failures where false alarms and malicious nodes could be disastrous to the network environment. This reputation system suffers from no such Byzantine failures, and is therefore an important advancement to previous collaborative worm containment systems.

The paper begins by providing a background into relevant work done in the areas of trust modeling, worm quarantine systems, and collaborative approaches to network security. It then continues by describing the trust model that we developed for use in this system, the simulator created to test the reputation system, and the experiments that we conducted to determine the reputation system's effectiveness in the face of extremely virulent scanning worms. The paper concludes by providing the results of the simulations, as well as a discussion of the results and directions of future work.

2. RELATED WORKS

The most notable use of recommendation and reputation systems in online applications is the Internet auction site eBay.com, where users give publicly viewable recommendations about experiences they have had with sellers and buyers. These recommendations are then used to gauge the seller or buyer's ability to perform transactions, possibly saving users from initiating a transaction with an unsavory character.

In addition to systems such as eBay, a number of more general frameworks have been developed. Abdul-Rahman and Hailes have created a number of systems for sharing and managing trust information in distributed environments [1][2][3]. Rather than directly adapting their systems, we chose to utilize the general ideas of trust mentioned in their work. In particular, they use decentralized, asymmetric trust between two distributed entities to characterize their distributed trust model. They also make a distinction between trust gained by first-hand experience and trust gained through the recommendation of others. These concepts have proved extremely useful in the development of our own distributed trust model.

There are also a number of models that take a less general approach to the problem of distributed trust. Some of these models are created for use in eCommerce and eBusiness situations and use various types of context information, such as the size of the transaction or the type of business in question, to account for specific shortcomings in the trust model [13][23][24]. Others make an attempt at a general framework, but also require this idea of context to tune their systems to different situations that may arise [4][6][16]. This use of context makes their system more adaptable to various special circumstances, but it also requires detailed management so that novel situations can be taken into account as the system develops and grows. Such management makes these types of systems impractical for use in our trust model where new technologies are developed daily.

With the ever increasing frequency of worm and virus attacks, it is no surprise that there have been a large number of research initiatives aimed at slowing or stopping the spread of this malware. A number of publications have aimed at the analysis of real and theoretical viruses and worms. The first of which was written by Spafford after the first known Internet worm in 1988 [18]. The analysis provides a historical perspective of the origins of worms, including the first theoretical use of worms in computer administration. There have also been in-depth analyses of recent worms, such as *Code Red* and *SQL Slammer* [9][11]. Staniford, Paxson, and Weaver also posited a number of theoretical worms that could wreak havoc on the Internet at-large [19]. Finally, the same authors along with Cunningham, provided a detailed introduction to worms and their propagation mechanisms, scanning methods, and payload [20].

There have also been a number of different approaches to preventing worm infections. A paper released by the United States' National Security Agency advocates the use of various orthogonal defenses, including firewall technologies, sandboxing, and stack-guarding, to stop self-propagating worms [5]. Some other approaches include automatically generating patches to fix vulnerabilities that worms exploit [17], throttling connections to detect and limit the impact of infected hosts [22], and an interesting approach where managed local area networks are automatically reconfigured to quarantine infected computers and still maintain availability of services [15].

Furthermore, there have been some approaches to preventing malicious activity that utilize collaboration among nodes. One such approach, named Pushback, aims at stopping denial of service and distributed denial of service attacks by throttling connections at high bandwidth upstream routers [8]. Another system developed for the quarantine of worms in enterprise networks organizes hosts into groups called cells and aims at preventing inter-cell infection by blocking traffic out of a compromised cell [21]. Worms are detected by counting the number of failed connections, and when the number of failed connections increases above a pre-set threshold, the cell is considered compromised and communications are blocked. Collaboration among cells allows this threshold to be reduced for quicker detection and prevention of worm activity. This threshold, however, can be reduced to such a point where false alarms cause a complete denial of service in the enterprise network. Nojiri et. al. discussed several general models for worm containment and their inherent benefits and drawbacks [12]. One important conclusion of that work is that collaborating organizations greatly decrease the reaction time of containment systems against worms, but an increased number of collaborating organizations is also detrimental to the functionality of the system in the case of false alarms. The authors of that paper suggest the use of a decay, or back-off, function to minimize the effects of these false alarms. All of the collaborative worm containment systems mentioned above provide useful features that aid in the quarantine of random scanning worms, but they also suffer from the same Byzantine failure where malicious nodes or false alarms can cause the entire system to fail, sometimes catastrophically.

3. TRUST MODEL

3.1 Overview

To create an Internet-scale system for the quarantining of random scanning worms, great care must be taken to protect against misuse and abuse. The single defining characteristic of this system is its use of a trust model in its operation. This use of the concept of trust among nodes in the network provides the system with the ability to utilize the recommendations of directly trusted and transitively trusted nodes while maintaining their autonomy.

To provide a fully automated, Internet-wide trust system, we borrowed heavily from our intuitions and experiences in various social situations, as well as concepts presented in previous trust management research. The general concept of our system is to view the various nodes on the Internet as autonomous beings with their own opinions of other nodes, just as humans have their own opinions and reputations. This view is generally consistent with that of the BGP system, where each autonomous system makes its own decisions on routes based on individual policy elements. We then include a system of recommendations where neighbors may provide their opinion of hosts based on events that have taken place in the past. Just as humans use the experiences of their friends and family to form opinions of people they have not yet met, so do the nodes in our system form opinions of hosts based on their experiences and the experiences of other nodes near them. Finally, just as humans trust people to a level defined by their degree of relation with them, so do the nodes in the reputation system. For a node to communicate with other, more distant nodes it must trust near-by neighbors to properly route traffic; therefore, our system places more strength in the recommendations of closer nodes than in more distant nodes.

In our system, we only consider malicious activity, worm traffic in specific, and therefore the reputation value in the system is a measure of how notorious an attacker is. The system springs into action when a victim node detects the malicious activity from the attacking host. This victim node then blocks the attacker and forwards a recommendation to all of its neighbors. These neighbors, individually, consider the recommendation and make changes to the reputation value they have stored for the attacker, indicating the attacker's disrepute. If any of these neighbors' reputation values rises above a certain threshold, the attacking host is blocked at that neighboring node, as well. The neighbors then propagate an updated recommendation to their neighbors until the strength of the recommendation is significantly reduced due to the distance that the recommendation has propagated from the victim. Over time, the reputation value for this attacker is decayed appropriately by each node individually, as long as the attacker's malicious activity has stopped. This process is repeated for every attack, and for every attacker, thereby creating localized reputation values at each node in the system for each attacker about which the nodes have received recommendations, or by which the nodes have been attacked directly.

3.2 Requirements

There are several requirements to ensure the fairness of this system. The first, and most obvious, requirement is that the victim node that is being attacked must block the attacker's traffic as soon as the attack is detected. The motivation for this requirement is quite intuitive – the victim should trust its own experiences implicitly. There are, of course, methods by which a determined attacker may spoof attacks from other hosts causing the victim node to erroneously block traffic from the spoofed host. Authentication mechanisms to prevent such misuse of the system, as well as combating spoofed recommendations are described in section 3.3.

The second requirement is that no one node should cause any other node to implement a block based solely on its recommendation without corroboration from other nodes. We must be careful to prevent recommendations from entering a loop and artificially increasing the reputation value at the nodes in that loop. Therefore we add a path attribute, similar to the AS-PATH attribute in BGP, which lists the autonomous systems that have forwarded the current recommendation. If the current node is in this path list, then the recommendation must be ignored as it has already been received and forwarded. Not only does this prevent loops, but it provides a sense of connectivity between nodes. For instance, if a particular node is highly connected to a victim node, then it is likely that the particular node will receive many recommendations from various paths, and therefore the reputation increase will be significantly greater than a single recommendation. The key, of course, is that this increase in received recommendations is linked directly to the connectivity between the node and the victim which originates the recommendation. Thus, the increased connectivity between the two nodes indicates increased probability of transiting each other's traffic, which translates to a greater overall increase in the reputation value for the attacker. This requirement of corroboration, along with the path attribute, prevents a Byzantine general scenario, where a small number of nodes can force an arbitrary node to block specific traffic without significant corroboration from additional sources. Yet, this requirement still allows for reputation increases that are intimately tied to the connectivity between the victim node and the nodes that receive the recommendations.

The third requirement of our system is that the reputations that each node holds should decay over time when no malicious activity is reported or detected. This allows a host to rejoin the network once the malicious activity has ceased, whether because of cleaning a worm from a computer, or because of an IP address change caused by the dynamic host configuration protocol (DHCP). If, however, the malicious activity continues, we also require that nodes that detect this activity and have already blocked the attacking node update their own reputation accordingly to ensure that the block remains in place. This requirement has the effect of creating a perimeter surrounding the offending node that it cannot penetrate. As a result, nodes outside of this perimeter may lower their reputation and stop blocking because the perimeter nodes take the responsibility of blocking. The longer attacks last and the more widespread they are, the further from the victims this perimeter gets until eventually it reaches the attacker, thereby totally cutting off all access to the Internet for the attacking host.

The final requirement is that the strength of the recommendation must diminish as the recommendation propagates further from the originating victim. Additionally, this propagation mechanism must cease when the strength of the recommendation becomes significantly lessened when compared to the benefit gained by continuing to propagate the recommendation. Essentially, we require that highly connected nodes continue propagating the recommendation even if it has lost significant strength, but lesser connected nodes should stop doing so when the strength is so low that the propagated recommendation would not have a significant impact on its neighbors. We diminish the strength of the recommendations with respect to the approximate amount of trust placed in a node to route data. Direct neighbors are more intimately tied with the task of routing data and therefore their recommendations should be given more weight than those who are less intimately involved, such as the nodes farther from the victim. This produces a hierarchy of trust loosely based on the proportion of data, on average, that a given near-by node will route. Hence, direct neighbors are most trusted because all data must flow through at least one of them, whereas more distant nodes may have a much smaller portion of the total data from a source node flowing through it. In the case of BGP autonomous systems, peering relationships provide a level of trust implied by the requirements placed on the routing of traffic by transit and customer autonomous systems. Therefore, we can use these peering relationships as our basis for transitive trust for other, indirect BGP autonomous systems. This transitive trust, as previously stated, is then modulated by the number of nodes involved in the trust, i.e. the number of nodes between the two endpoints of the transitive trust. This notion provides an accurate depiction of the level of trust placed on a given node with respect to routing data. Combining this decrease in strength with the path attribute that we previously described, provides an excellent indicator of the trust between two autonomous systems based on the rough percentage of data that they transit for each other, as described by their connectivity and distance.

3.3 Description

The reputation system's trust model is described by a set of rules and an equation that helps to fulfill the requirements previously discussed. The details of this model are given in Figure 1. The most important part of the model is the formula for determining how to adjust the reputation score of an attacker when recommendations are received. This formula is given as Equation 1 in the figure below. There is also a rule that determines when a node should block a particular attacker's traffic, given as Rule 1 below. Clearly, these two elements satisfy the first of our requirements that the victim node that is being attacked must block the attacker immediately. The distance from the victim to the current node in Equation 1 is zero since the victim and current node are the same. Therefore, a value of one is immediately added to the reputation for the attacker, hence blocking it based on Rule 1.

This equation and rule combination also satisfies our second requirement that no small number of nodes should cause any other node to block another node without corroboration, thereby preventing Byzantine general vulnerabilities. If we consider recommendations from direct neighbors, where the distance from the victim to the current node is one, then the combination of Equation 1 and Rule 1 will require a simple majority of the current node's direct neighbors to also provide recommendations about the same attacker for a block to be imposed. Even in the situation where a node has only one direct neighbor, a single recommendation from that neighbor can not cause the node to block; corroboration is required either in the form of a direct attack or recommendations propagated from more distant nodes. Therefore, this system does not suffer from the ill effects of false alarms or malicious communications like previous attempts at collaborative worm containment systems.

It is also possible for some combination of recommendations from more distant nodes to be received by the current node and to cause a block without any recommendations from direct neighbors. Thus, the system allows recommendations from both directly trusted and transitively trusted nodes to be counted towards the reputation of an attacker, but certainly more than a single report of the misbehavior of the attacker is required for blocking.

To provide a mechanism for the decay of reputation over time, we define discrete time periods at which the reputation is decayed, known as the λ periods. We use Rule 2, given in the Figure 1, to decay the reputation for the attackers stored at the current node by a percentage of the total reputation. This decay is based on the number of neighbors for that node. Furthermore, we differentiate the discrete λ periods from the time at which the decay is actually evaluated, known as the management period. This management period allows the administrators of the autonomous systems to better control the minimum time that a block will be in place. For instance, an administrator might want blocks on his autonomous system to last at least ten minutes, but would like the actual effective length of the reputation to be much less, perhaps only a minute. This reduces the computational requirements of the evaluation of the decay to a minimum time period, as defined by the management period, but also allows for quick turnover of reputation information. Moreover, this decay allows nodes previously marked as malicious to rejoin the network after they have stopped their attacks for a number of λ periods commensurate with their reputation rating. Additionally, when a node blocks traffic from an attacker due to the attacker's reputation by Rule 1, that node should inspect the contents of the traffic that has been dropped to see if the attacker is continuing the malicious behavior. If the traffic is found to be malicious, then that node should add one to the reputation value every management period. This ensures that not only will the attacker remain blocked, but that the attacker's block at that node will actually last longer due to continued worm attacks. In effect, this ensures that only the foremost blocking node must deal with maintaining reputation for the attacker, thereby creating a quarantine perimeter, while other nodes behind the foremost blocking node can safely decay the attacker's reputation without having to worry about maintaining the block.

Finally, to fulfill the fourth requirement, we must use the third rule in Figure 1 to limit the distance that the recommendation can travel. Since the distance affects the strength of the recommendation, as given in Equation 1, we do not want the strength to be reduced so much that it takes N^2 distant recommendations to be equivalent to one recommendation from a neighbor, where *N* is the neighborhood of the node as defined in Figure 1. Recommendations with such a small strength are essentially meaningless in our model. Furthermore, we want to make sure that highly connected nodes continue to forward the recommendation to other lesser connected nodes, so we use the neighborhood size to determine if the recommendation propagation should continue. It is also clear by examining Equation 1 that as the distance increases, the strength of the recommendation is reduced multiplicatively.

Practically, the equation and rules given in Figure 1 are not easy to implement exactly as they are shown. Equation 1 can be implemented by initializing an array of reputations to zero and assigning new attackers to empty spots in this array, possibly with a collision resistant hashing function. At the receipt of additional recommendations, the previous value of the attacker's array entry is used and the new recommendation's value is added to it and stored in the same array location. Blocking of the attacker is simply implemented by checking the value of the new reputation after a recommendation value is added to see if it is greater than one. We implement decay of the reputation by iterating through the array of reputations and reducing each reputation multiplicatively by a factor of

$$\left(\frac{\left\lfloor \frac{N}{2} \right\rfloor}{\left\lfloor \frac{N}{2} \right\rfloor + 1}\right)^{\frac{m}{\lambda}}$$

at the expiration of the management period; where m is the length of the management period, N is the neighborhood size, and λ is the length of the decay period. The attacker is unblocked when the decayed reputation value becomes less than one, and the attacker is removed from the array when their reputation value is considerably smaller than one. The propagation limit is trivially implemented by checking for the number of hops traveled up to the current node and comparing the value with Rule 3 above. A truly efficient implementation of this system is of utmost importance and should be investigated in detail, but defining such an implementation is beyond the scope of this paper.

Equation 1:
[Evaluated at receipt of a recommendation]

$$rep(a,0) = 0$$

$$rep(a,t) = rep(a, prev(a,t)) + \frac{1}{\left(\left\lfloor \frac{N}{2} \right\rfloor * dist(v(a,t))\right) + 1}$$

Rule 1:
[Evaluated after Equation 1]

$$if\left(rep(a,t) \geq \left(1 + \frac{1}{\left\lfloor \frac{N}{2} \right\rfloor}\right)^{\left\lfloor \frac{t}{m} \right\rfloor * \left(\frac{m}{\lambda}\right)} \right)\{block(a)\}$$

Rule 2:
[Evaluated at the expiration of the management period]

$$if\left(rep(a,t) < \left(1 + \frac{1}{\left\lfloor \frac{N}{2} \right\rfloor}\right)^{\frac{t}{\lambda}} \right)\{unblock(a)\}$$

Rule 3:
[Evaluated after Rule 1]

$$if\left(dist(v(a,t) \leq (N-2)\right)\{propagate\}$$

a – Attacker, *t* – Current time, λ – Length of the decay period
m – Length of the management period
N – Size of the current node's neighborhood (# neighbors + 1)
rep(x,y) – Reputation value for attacker x at time y
dist(z) – Distance from the current node to victim z
v(x,y) – Latest victim to send a recommendation for attacker x before time y
prev(x,y) – Time of the recommendation about attacker x before time y

Figure 1: Trust model equation and rules

Thus far we have shown that the trust model itself is fair with regard to the requirements that we described in section 3.2, but we have not yet addressed the security of the system against spoofed attacks, faked recommendations, or alteration of the attributes contained in the recommendation messages. The first problem that needs to be dealt with is the use of spoofed addresses in attacks. If the actual attacker were to use a spoofed IP address, rather than its own, it would cause the system to inadvertently distribute a recommendation about the spoofed address, perhaps even causing this spoofed address to be unduly blocked at a number of autonomous systems. This particular problem is quite vexing. A solution to such a problem can not simply be built into our reputation system alone, as it is a much wider problem with the Internet architecture in general. The typical solution to this problem is the use of ingress filtering at the border router of service providers to ensure that only properly allocated IP addresses are used by their customers. However, evidence of continued spoofing throughout the Internet indicates that there are a large number of service providers that do not follow this practice. One interesting solution to this problem is the use of IP Traceback schemes which validate the path that packets take to determine the actual sender of the packets, rather than simply relying on the address given in the packet's header [14].

Another attack, targeted specifically to our reputation system, could fake recommendation messages. With such an attack, an attacker that has taken control of a node's reputation system can send any number of faked recommendations about any address, and make it seem like it has originated from just about any other participating node. Clearly, if an attacker is able to do this, then they have free reign on implementing blocks for any arbitrary address simply by issuing a suitable number of faked recommendations. Our trust model builds some defense against this by limiting the distance that a given recommendation can propagate, so even if the subverted node sends out fake recommendations that it has been attacked, these recommendations only traverse a finite number of links. This ensures that the effects of these faked recommendations are limited to a localized subset of the total nodes participating in the reputation system. Additionally, if a fixed λ period is chosen, then we can safely assume that a participating node that is following the rules laid out in section 3.2 will not be able to send more than one recommendation for a given attacker per λ period because they should be blocking that attacker. Furthermore, if only one, or a small number, of near-by subverted nodes are sending recommendations once per λ period, then the decay of the reputation added by these recommendations will make it extremely difficult to cause wide-spread blocking of an arbitrary address. We can also consider the use of digital signatures or cryptographic hash functions to verify the originator of a given recommendation. Moreover, we may even validate the attribute information for the recommendation with the digital signature and ensure that these attributes have not changed. These digital signature schemes require significant public-key infrastructure, however, which is a substantial hurdle to their use in the Internet at-large. We may also consider using the previously mentioned IP Traceback mechanisms for verification of the recommendation origin. These security concerns are quite serious, but investigation of specific methods for their implementation in this system is not the primary focus of this paper and so we will leave in-depth review of these possible mechanisms to future research.

4. WORM CONTAINMENT SIMULATION

4.1 Simulator Description

To properly simulate the unique response of our reputation system to random scanning worms, we created a custom network simulator to allow for maximum control over the implementation of our trust model. The simulator is a typical discrete event simulator utilizing an event queue. Messages, or packets, carrying pertinent information pass between events. Each event represents a specific network event in our system, such as routing, receiving an attack, or propagating a recommendation. The simulator begins by reading an input file and creating a number of node entities. After the configuration information is read, the simulator creates a routing table for each node, and chooses a number of nodes, at random, that are vulnerable to the worm. These vulnerable nodes are equivalent to having all hosts in an autonomous system vulnerable to worm attack. The simulator then begins propagation of our worm, described below, from an originating node. The initially infected node will choose, at random, a number of nodes corresponding to the worm's scanning rate and send worm attacks that will be routed according to the routing table and ultimately received by the victim. If the victim has been determined to be vulnerable, then the victim will also begin spreading the worm to randomly chosen nodes at the worm's scanning rate. When a node becomes compromised by worm attack, we assume all hosts in the representative autonomous system are compromised. Furthermore, these attacks may be blocked at intermediate routing nodes according to the trust model simply by dropping the traffic from the attacker. When a node gets attacked, we assume that the detection mechanism, whatever that might be, is one hundred percent accurate and begins the recommendation propagation process outlined above. This may seem onerous, but the purpose of this initial work is to determine the best possible performance of the system against various types of very virulent worms and to better understand its unique ability to utilize trust in quarantining Internet worms. Additionally, all nodes have the reputation system running, so at preset periods the reputation management process will be run on the routers based on the management period length, which is defined globally in this simulator. These events will continue until a specified length of time is completed when the simulator will gather statistics about the simulation.

4.2 Experiment Description

A number of parameters were used to test the effectiveness of our system in halting the spread of a malicious Internet worm. These parameters included the percentage of nodes in the network that are vulnerable to attack, the scanning rate of the worm, and the λ period of the system. By altering these parameters we can test the system's reaction to extremely virulent worms and understand the effects of the various parameters on the worm's propagation. We ran twenty trials for each specific scenario to minimize the effects of randomization on the results gathered. Each of these trials lasted for one hour of simulated time. At the end of these trials, the simulation provides the number of vulnerable nodes that were compromised by the worm and the last time an attack reached its intended destination.

The network which was used in this experiment consisted of ten identical sub-networks of one hundred nodes each that are connected together by three backbone nodes. These nodes represent BGP autonomous systems, connected to one another similarly to the way autonomous systems are connected in the Internet per the Internet Mapping Project [7]. The percent of vulnerable nodes was set to one, five, or ten percent of the total network size, plus the original infected node. The number of nodes this percentage represented was rounded down to the nearest integer in cases where the percentage was not a whole number. We also tested different scanning rates of ten and one hundred nodes scanned per second to determine the reputation system's reaction to increased scanning rates.

It is also very important to connect our results to real worms, as well as other research efforts that attempt to place a maximum bound on the effectiveness of worm containment systems. As a means of comparison, both versions of the *Code Red* worm had a vulnerable population of only approximately 0.0084% of the entire Internet address space [9], while the Slammer worm had an even smaller population of only 0.0018% [11]. Additionally, Moore et. al. [10] provided a rating for how well a worm containment system performs by using the percentage of the vulnerable population that becomes compromised as an indicator of effectiveness. Their work suggests that if less than ten percent of the vulnerable population becomes compromised, then the worm has been well contained, ten to ninety percent indicates partial containment, and greater than ninety percent indicates an uncontained worm. Finally, with use of the epidemiological model popularized by Moore et. al. [10], and Scandariato and Knight [15], Figure 2 shows that after ten seconds our least virulent simulated worm is many orders of magnitude more virulent than *Code Red v2* due to the increase in the vulnerable population. With our worm having the same scanning rate as *Code Red v2* and a vulnerable population of one percent, after only ten seconds our worm has compromised 23% of the vulnerable hosts when unimpeded compared to only 0.00028% of the population for *Code Red v2*.

$$x(t) = \frac{K * e^{\frac{s*N}{T}*t}}{N + K * e^{\frac{s*N}{T}*t}}$$

$$K = \frac{N * H}{N - H}$$

For Code Red v2:
N=360,000, H=1, T= 2^{32} -1, s=10, x(10)=0.00028%

For simulated worm, large (1,003 node) network:
N=11, H=1, T=1,003, s=10, x(10)=23.0%

x(t) - Percentage of vulnerable population at time t
N - Number of vulnerable nodes (including initially infected)
H - Number of initially infected nodes
T - Number of nodes in the network
s - Number of nodes scanned per second
t - Current time

Figure 2: Epidemiological model comparison of simulated worm to Code Red v2

5. RESULTS

We begin the discussion of our results by comparing the epidemiological model provided in section 4.2 with the actual propagation of the worm in our network simulator. This comparison will allow us to assess how faithful our worm simulation is to the mathematical model for random scanning worms provided in the previous section. As seen in Figure 3, the simulated worm propagation is almost an exact match for the graph of the worm propagation based on the epidemiological model. There is a slight shift in the graph of the simulated worm when compared to the epidemiological model that is caused by the link delay that is present in our simulator. This link delay is applied on traversal of each link in the simulation and therefore the routing of attacks and recommendations is subject to it. The epidemiological model, however, assumes instantaneous infection and therefore this accumulated delay in our simulation causes the shift shown.

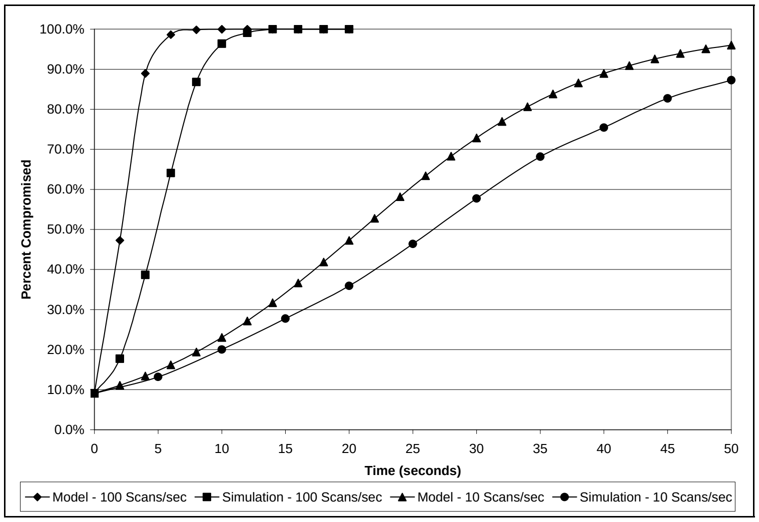

Figure 3: Comparison of epidemiological model to simulated worm propagation

Now that we have established that the simulator propagates worm attacks realistically in our test network, we can investigate the effects of the λ period on the number of vulnerable nodes compromised and on the last time at which an attack was received by its intended victim. For the purposes of clarity, we remove the node that is initially infected with the worm from the vulnerable population, and therefore only depict the number of new infections the worm produces. Intuitively, the length of the λ period determines how long the reputation lasts at a particular node, so as the λ period value increases so does the length of the block. This should decrease the number of compromised nodes as well as the time the last attack was received. Because the management period parameter is solely defined by the administrator of the autonomous system based on the needs of their site, we fix this parameter to sixty seconds. Additionally, we fix the percentage of vulnerable nodes to one percent and the scanning rate of the worm to ten scans per second. To truly understand the usage of this parameter, we must examine the minimum boundary where the λ period is set low enough so as to make the system totally ineffective, and then examine how increasing this value affects the propagation of the worm. We begin our evaluation with a λ period value of 2, which, as Figures 4 and 5 show, makes the system completely ineffective because the reputation that the various recommendations provided is decayed almost immediately. As we increase the λ period value, however, we see that both the time of last attack and the percentage of compromised nodes drop until they reach a plateau at six seconds. The slight variation between results from six to ten seconds is clearly caused by the random nature of the scanning worm and the simulation. Upon closer investigation, this plateau indicates a boundary for the effectiveness of the system. This boundary indicates that these vulnerable nodes were compromised before the reputation system could spread enough recommendations to completely quarantine the worm. Existence of such a boundary is to be expected as the reputation system acts as a sort of ‘anti-worm’ that reacts locally to the worm attack and only has global effect after widespread attacks have occurred.

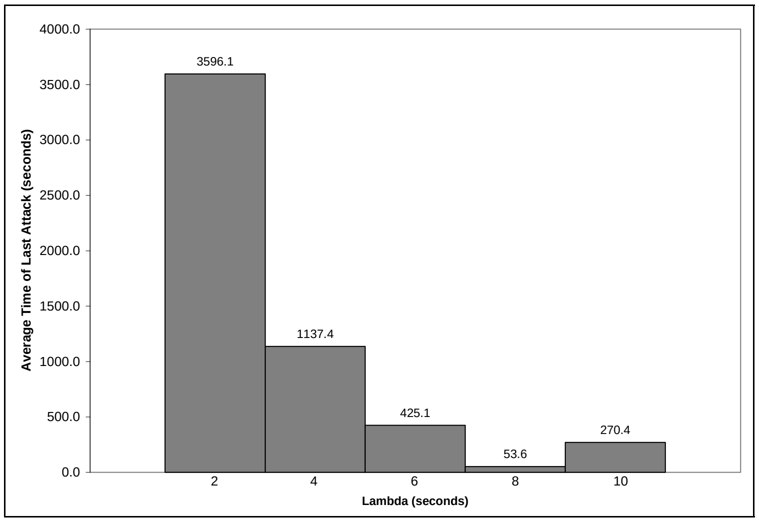

Figure 4: Comparison of λ period effects on average time of last attack

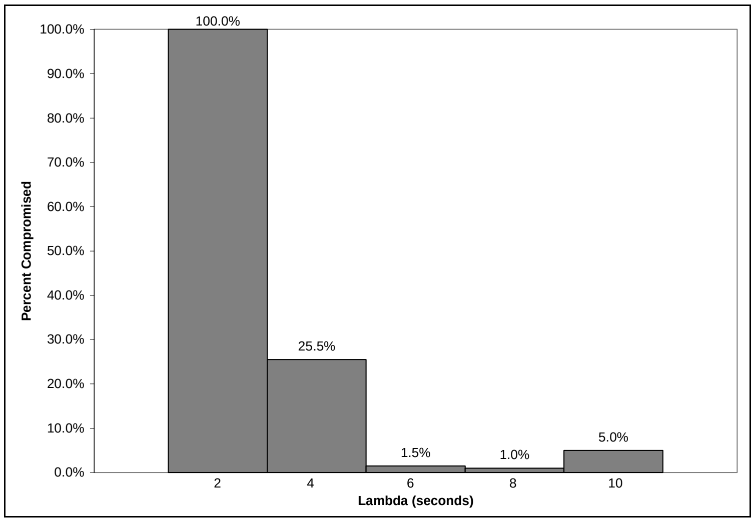

Figure 5: Comparison of λ period effects on percentage of vulnerable nodes compromised

We can now utilize our understanding of the λ period to study the reputation system's ability to quarantine worms of differing virulence. Since we have shown that any λ period value above six seconds yields similar system performance, we choose to fix the value at sixty seconds to match the management period value. This indicates that every time the management period expires, a full decay is performed on the reputations. As before, we fix the scanning rate of the worm to ten scans per second. We can now measure the performance of our system on worms with one, five, and ten percent vulnerable population sizes as described in section 4.2. It is quite clear that increased vulnerable populations should increase the chances of a given worm scan successfully finding and compromising a node, and thus the percentage of compromised nodes should increase accordingly. Additionally, we should also see that the time that the last attack was received should increase as there are more compromised hosts for which the system needs to issue recommendations before blocking. Figures 6 and 7 show that increased virulence leads to an increase in compromised nodes as well as an inability of the system to completely stop worm traffic. It is interesting to note, however, that the system is able to effectively quarantine even the five percent vulnerable worm in a reasonable amount of time.

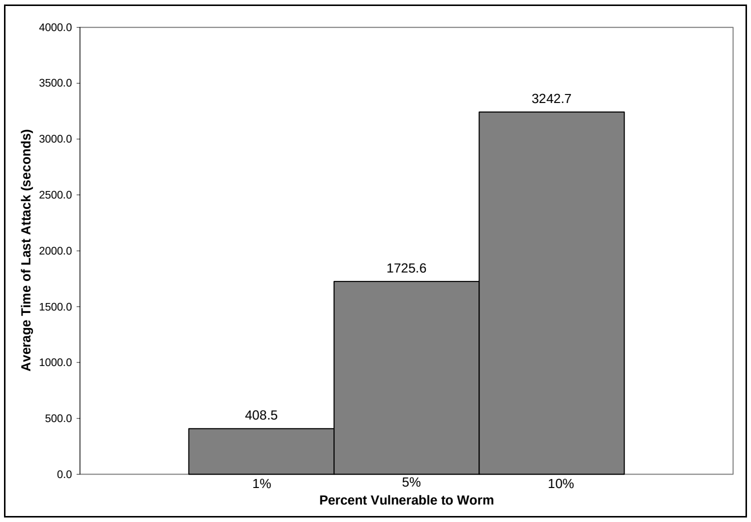

Figure 6: Comparison of worm virulence effects on average time of last attack

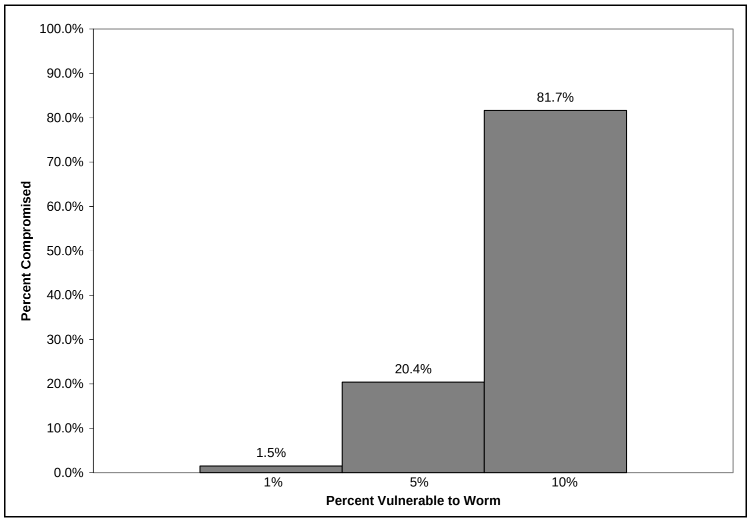

Figure 7: Comparison of worm virulence effects on percentage of vulnerable nodes compromised

Finally, we compare the reaction of our reputation system to different scanning speeds. We again fix our management and λ periods to sixty seconds, and the percentage of vulnerable nodes is again set to one percent. We now test the reaction of our system to a ten scan per second worm and a one hundred scan per second worm. Since our system is completely reactionary to the attacks, the quicker the attacks occur, the quicker our system should send out recommendations in response to those attacks. Figures 8 and 9 show that our system does indeed react more quickly to the increased scanning speeds and, in fact, quarantines it much faster. We also see that the percentage of compromised nodes increases slightly as the scanning rate increases. This is consistent with our previous assertion that these nodes are scanned and compromised before the system forces global blocking of the compromised nodes. In Figure 10 we can see the drastic effect that the reputation system has on the spread of these two worms when compared to their propagation in the unprotected network.

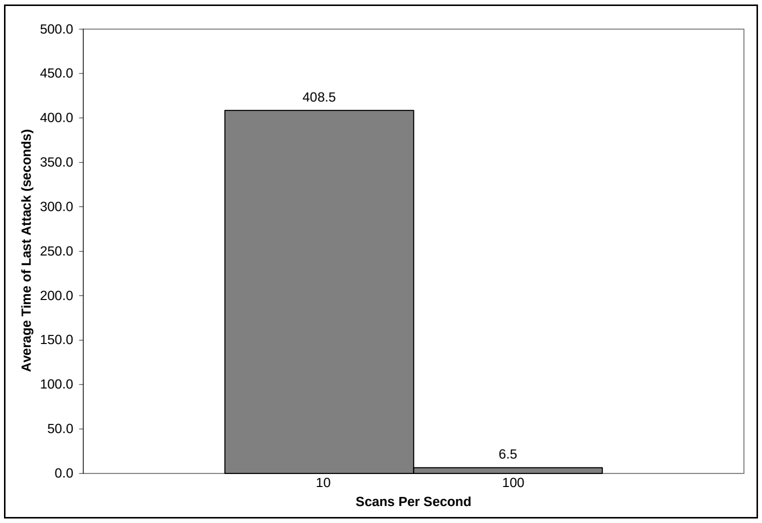

Figure 8: Comparison of scanning rate effects on average time of last attack

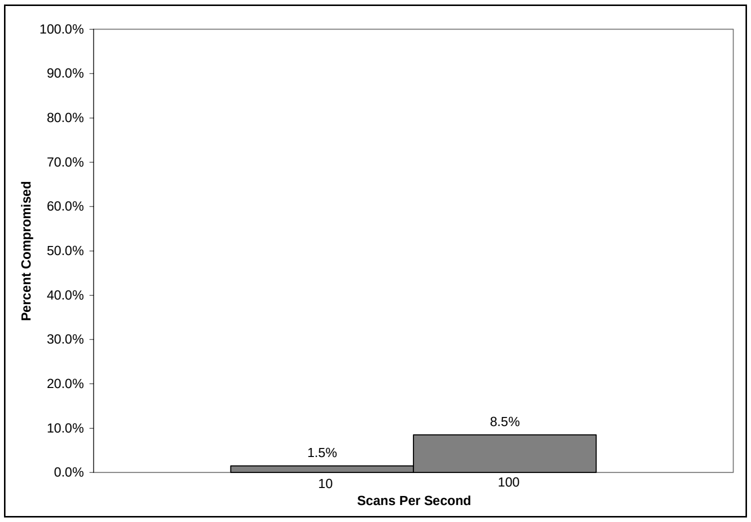

Figure 9: Comparison of scanning rate effects on percentage of vulnerable nodes compromised

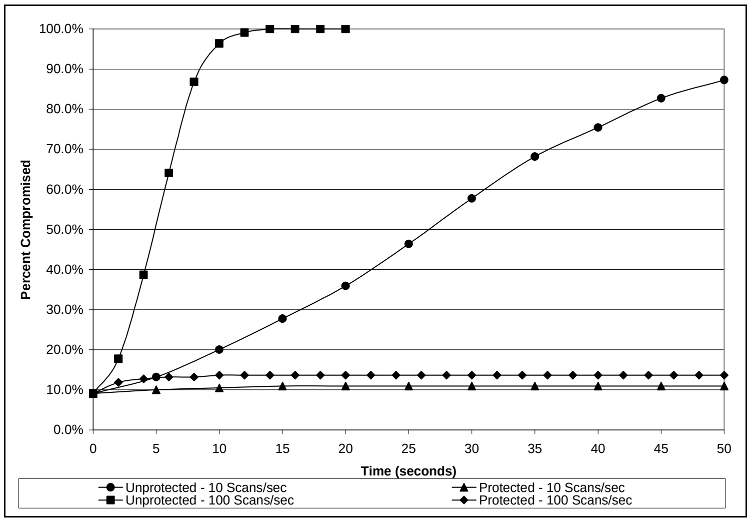

Figure 10: Comparison of worm propagation in unprotected network versus protected network

6. CONCLUSION

In developing this paper, we wanted to focus on gaining a basic understanding of the dynamics of our reputation system in the face of various worms, as well as setting a bound for the best possible performance of the system. The results given in the previous section demonstrate that our system is able to effectively quarantine worms with varying scanning rates, and with extremely large vulnerable populations – up to five percent. We also noted that previous worms released into the Internet did not have a vulnerable population nearly the size of the worms against which we tested our system This gives us extremely encouraging results that lead us to believe that even with less than ideal conditions, like reduced detection rates and less participation, the system could be extremely effective in quarantining worms in the Internet at-large. Furthermore, our results show that the ten and one hundred scan per second worms with a one percent vulnerable population are considered well-contained by Moore et. al., while the ten scan per second worm with a five percent vulnerable population is partially contained [10]. Given the results, we can also infer that if a scanning rate greater than one hundred scans per second were used, the system would react even faster and only allow a slightly larger number of vulnerable nodes to become compromised.

While we have shown that this system is more than capable of quarantining the malicious behavior of a scanning worm, it can also be used to quarantine other types of malicious behavior that is widespread throughout the Internet, due to the unique localized reactive nature of the system and its total independence from malware detection technology. Such malicious behaviors may include, but are not limited to, spam and human-propagated computer viruses. These malicious behaviors would simply need a suitable, customized λ period value that is commensurate with the speed at which they propagate. As a byproduct of the blocking of this widespread misbehavior, our system also limits the amount of total bandwidth used in propagating the virus, worm, or spam. This can be especially useful in situations where ongoing, high volume worm traffic effectively shuts down major links in the Internet infrastructure as *Slammer* did [11]. However, because this system relies on the widespread detection and reporting of malicious behavior, it will do little to stop many-to-one or one-to-one attacks. The system is based on the contention that a single host may attack many nodes, and that sharing the attack information in a way humans share recommendations allows for quicker and more effective responses where needed. In the case of the many-to-one or one-to-one attacks such as distributed denial of service or denial of service attacks, this principle is reversed. This also means that our system would be ineffective at stopping the so-called flash and hit-list worms that propagate by using a previously created list of vulnerable hosts because there is no ongoing, widespread malicious activity on which recommendations could be based [19]. The most distinctive feature of our system is its resilience to Byzantine failures that are detrimental to other collaborative worm containment systems.

The novelty of this system leads to many avenues of future research which will be instrumental in establishing its viability as a method for securing the Internet from such widespread abuse as viruses, worms, and spam. An investigation of the specific implementation requirements and an efficient design for the system are required. Along with this, continued research needs to be done to pinpoint security mechanisms, such as digital signature schemes, which will satisfy the security requirements discussed in section 3.2. Study of the interaction between our reputation system and the BGP routing protocol should also be conducted to determine the effects of the blocks on the routing topology, as well as to determine the possibility of combining the two protocols to complement one another. Finally, detailed studies of the dynamics of the propagation, as well as the effects of detection and participation rates on containment, are necessary to derive a complete model of this system's implementation.

REFERENCES

[1] ABDUL-RAHMAN, A. AND HAILES, S. *A Distributed Trust Model.* Proceedings of the 1997 Workshop on New Security Paradigms, Langdale, Cumbria, United Kingdom, 1997. ACM Press, Pages: 48-60.

[2] ABDUL-RAHMAN, A. AND HAILES, S. *Using Recommendations for Managing Trust in Distributed Systems.* Proceedings of the IEEE Malaysia International Conference on Communications, Kuala Lumpur, Malaysia, 1997.

[3] ABDUL-RAHMAN, A. AND HAILES, S. *Supporting Trust in Virtual Communities.* Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, 2000.

[4] ABERER, K. AND DESPOTOVIC, Z. *Managing Trust in a Peer-2-Peer Information System.* Proceedings of the 10th International Conference on Information and Knowledge Management, Atlanta, GA, 2001. ACM Press, Pages: 310-317.

[5] ALBANESE, D., WIACEK, M., SALTER, C., AND SIX, J. *The Case for Using Layered Defenses to Stop Worms.* Viewed at: www.nsa.gov/snac/support/WORMPAPER.pdf on November 15, 2004.

[6] CHEN, R. AND YEAGER, W. *Poblano: A Distributed Trust Model for Peer-to-Peer Networks.* Sun Microsystems' JXTA project. Viewed at: http://www.jxta.org/docs/trust.pdf on June 2, 2003.

[7] CHESWICK, W. AND BURCH, H. *Internet Mapping Project.* Viewed at: http://research.lumeta.com/ches/map/ on October 17, 2004.

[8] IOANNIDIS, J. AND BELLOVIN, S. M. *Implementing Pushback: Router-Based Defense Against DDoS Attacks*. In Proceedings of the Network and Distributed System Security Symposium (NDSS), February 2002.

[9] MOORE, D., SHANNON, C., AND CLAFFY, K. *Code-Red: A Case Study on the Spread and Victims of an Internet Worm.* Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurement, Marseille, France, 2002. ACM Press, Pages: 273-284.

[10] MOORE, D., SHANNON, C., VOELKER, G.M., AND SAVAGE, S. *Internet Quaratine: Requirements for Containing Self-Propagating Code.* Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE, Volume: 3, Issue: 30 March-April 2003. Pages: 1901–1910.

[11] MOORE, D., PAXSON, V., SAVAGE, S., SHANNON, C., STANIFORD, S., AND WEAVER, N. *Inside the Slammer Worm.* Security & Privacy Magazine. IEEE, Volume: 1, Issue: 4, July-Aug. 2003. Pages: 33–39.

[12] NOJIRI, D., ROWE, J., AND LEVITT, K. *Cooperative Response Strategies for Large Scale Attack Mitigation*. Proceeding of the DARPA Information Survivability Conference and Exposition, 2003. Volume: 1, Pages: 22-24 April 2003. Pages:293–302.

[13] PIERRE, G. AND VAN STEEN, M. *A Trust Model for Peer-to-Peer Content Distribution Networks.* Viewed at: http://www.cs.vu.nl/pub/papers/globe/trustp2pcdn.pdf on June 2, 2003.

[14] SAVAGE, S., WETHERALL, D., KARLIN, A., AND ANDERSON, T. *Network Support for IP Traceback.* ACM/IEEE Trans. on Networking, vol. 9, no. 3, pp. 226–239, June 2001.

[15] SCANDARIATO, R. AND KNIGHT, J.C. *The Design and Evaluation of a Defense for Internet Worms.* Proceedings of the 23rd IEEE International Symposium on Reliable Distributed Systems, 2004, 18-20 Oct. 2004. Pages: 164–173.

[16] SHROBE, H. AND DOYLE, J. *Active Trust Management for Autonomous Adaptive Survivable Systems*. Proceedings of the 1st International Workshop on Self-Adaptive Software, Oxford, United Kingdom, 2000. Springer-Verlang, Pages: 40-49.

[17] SIDIROGLOU, S., AND KEROMYTIS, A. *Countering Network Worms Through Automatic Patch Generation.* Columbia University Technical Report, CUCS-029-03.

[18] SPAFFORD, E. *The Internet Worm Program: An Analysis.* Purdue Technical Report CSD-TR-823.

[19] STANIFORD, S., PAXSON, V., AND WEAVER, N. *How to 0wn the Internet in Your Spare Time*. Proceedings of the 11th USENIX Security Symposium, 2002.

[20] WEAVER, N., PAXSON, V., STANIFORD, S., AND CUNNINGHAM, R. *A Taxonomy of Computer Worms*. Proceedings of the 2003 ACM Workshop on Rapid Malcode, Washington, D.C., 2003. ACM Press, Pages: 11-18.

[21] WEAVER, N., STANIFORD, S., AND PAXSON, V. *Very Fast Containment of Scanning Worms.* Proceedings of the 13th USENIX Security Symposium. Pages: 29-44.

[22] WILLIAMSON, M. *Throttling Viruses: Restricting Propagation to Defeat Malicious Mobile Code.* Proceedings of the 18th Annual Computer Security Applications Conference, 2002. Pages: 61-68.

[23] XIONG, L. AND LIU, L. *Building Trust in Decentralized Peer-to-Peer Electronic Communities.* Proceedings of the 5th International Conference on Electronic Commerce Research, Montreal, Canada, 2002.

[24] XIONG, L. AND LIU, L. *A Reputation-Based Trust Model for Peer-to-Peer eCommerce Communities.* Proceedings of the IEEE Conference on E-Commerce, Newport Beach, CA, 2003.