

# On the Development of an Internetwork-centric Defense for Scanning Worms

Scott E. Coull  
Department of Computer Science  
University of North Carolina  
201 South Columbia Street  
Chapel Hill, NC 27599, USA  
scoull@cs.unc.edu

Boleslaw K. Szymanski  
Department of Computer Science, Lally 204  
Rensselaer Polytechnic Institute  
110 Eighth Street  
Troy, NY 12180, USA  
szymansk@cs.rpi.edu

## Abstract

*Studies of worm outbreaks have found that the speed of worm propagation makes manual intervention ineffective. Consequently, many automated containment mechanisms have been proposed to contain worm outbreaks before they grow out of control. These containment systems, however, only provide protection for hosts within networks that implement them. Such a containment strategy requires complete participation to protect all vulnerable hosts. Moreover, collaborative containment systems, where participants share alert data, face a tension between resilience to false alerts and quick reaction to worm outbreaks.*

*This paper suggests an alternative approach where an autonomous system in an internetwork, such as the Internet, protects not only its local hosts, but also all hosts that route traffic through it, which we call internetwork-centric containment. Additionally, we propose a novel reputation-based alerting mechanism to provide fast dissemination of infection information while maintaining the fairness of the system. Through simulation studies, we show that the combination of internetwork-centric containment and reputation-based alerting is able to contain an extremely virulent worm with relatively little participation in the containment system. In comparison to other collaborative containment systems, ours provides better protection against worm outbreaks and resilience to false alerts.*

**Keywords:** Network security; Computer worms; Malware protection; Worm containment; Reputation; Collaborative network defense.

## 1 Introduction

Internet worms propagate at a rate that makes manual intervention ineffective. Several studies suggested that Internet worms found in the wild can be best contained through the use of automatic containment mechanisms [8, 15, 9, 10]. In response to these observations, several automated containment systems were proposed. Generally, these systems are either *stand-alone* [14, 16, 20, 19, 2], where alerts are generated only from local worm detection mechanisms, or *collaborative* [1, 4, 11], where a group of possibly untrusted entities agree to share detection alerts. Stand-alone containment systems are clearly at a disadvantage when detecting a worm outbreak since they must rely on direct observation of the attack on their local network before enacting countermeasures, whereas collaborative systems can preemptively protect themselves using the observations of others. Moreover, previously proposed containment systems, both collaborative and stand-alone, implement countermeasures (e.g., blacklisting) in such a way that only hosts within participating networks are protected from infection. In addition, collaborative systems are susceptible to the effects of false alerts, caused by false positives of the underlying intrusion detection systems or malicious entities, which may trigger the implementation of countermeasures throughout the containment system. Thus, there is a tension between the speed with which collaborative systems react to worms and their resilience to false alerts.

In this paper, we explore an alternative approach to collaborative worm containment that attempts to provide a more scalable solution by avoiding localized containment. Specifically, we create a collaborative worm containment system within an internetwork of untrusted autonomous systems, such as the Internet. The subset of autonomous systems that participate

in the containment system form an overlay network and share alerts about worm activity, which are transmitted using the existing routing infrastructure. Rather than implementing countermeasures that only block worm activity destined for the local network, the autonomous systems implementing our worm containment system block *all* worm traffic passing through it, including traffic being routed to other autonomous systems. This implementation of *internetwork-centric containment* allows our worm containment system to protect a much larger portion of the internetwork, including autonomous systems that do not participate in the containment system. In fact, previous work has shown that such a strategy works well against virulent worms [21, 10].

Of course, since our collaborative containment system operates among untrusted parties, the system must be resilient to both ambient false alerts and malicious participants. This is especially true for any system based on internetwork-centric containment, which might be misused to perform denial of service attacks. Thus, we create a novel *reputation-based alerting* mechanism to maintain a balance between containment of virulent worms and resilience to false alerts. In essence, a participant in the system will broadcast an alert about an infected host through the containment system's overlay topology, and the alert will be weighted in proportion to the distance it has traveled when received by other participants. The participating autonomous systems accumulate the weighted alerts for each infected host as a reputation, and when the reputation for a particular infected host reaches a predefined threshold the autonomous system implements countermeasures to drop all traffic from that host temporarily. By considering the reputation threshold to be a weighted vote among participating autonomous systems, wherein the majority of participants must corroborate the attacker's malicious activity, we prevent false alerts or small groups of malicious participants from affecting the other participants in the containment system.

Our internetwork worm containment system is evaluated via simulation of a scanning worm that is significantly more virulent than any worm found in the wild to date. The results of our evaluation show that our worm containment system can effectively contain the extremely virulent worm with relatively little participation in the containment system, allowing less than 0.1% of vulnerable hosts to become infected with only 30% of all autonomous systems participating. We also provide comparisons to previous collaborative containment systems, which show that our combination of internetwork-centric countermeasures and reputation-based alerting provide a good balance between false alert resilience and containment. Finally, we explore the impact of real world conditions, such as packet delay or worm detection rates, on the efficacy of our containment system.

## 2 Related Work

There have been a variety of approaches designed to prevent worm infections within an enterprise network. Methods have been proposed to automatically patch vulnerable hosts [16], throttle scans from infected machines at gateway routers [20, 2], and automatically reconfigure local area networks to block communications from infected machines [14, 19]. These stand-alone worm containment methods, however, all suffer from an inability to perform preemptive countermeasures. This causes stand-alone containment systems to have a drastically slower reaction time to worm attacks. Additionally, such local countermeasures require fairly wide deployment throughout the networks of the internetwork to affect the global propagation of a worm in any meaningful way. In fact, a study by Moore et al. has shown that in order to effectively protect against a worm outbreak, defense measures must be both automatic *and* collaborative [10].

The slow reaction time of stand-alone containment mechanisms has been addressed by introducing mechanisms to share alerts among a number of cooperating organizations. Nojiri et al. provide a system that dramatically reduces the reaction time to worms by sharing alerts among a predetermined subset of participating nodes [11]. The system proposed by Kannan et al. instead reports the infection of local hosts to other organizations implicitly with a marker in the header of traffic from infected hosts, or explicitly via alert messages [4]. Both the Nojiri et al. and Kannan et al. systems utilize a push model for alerting. While this model provides excellent reaction to fast scanning worms, it can also lead to poor resilience to false alerts.

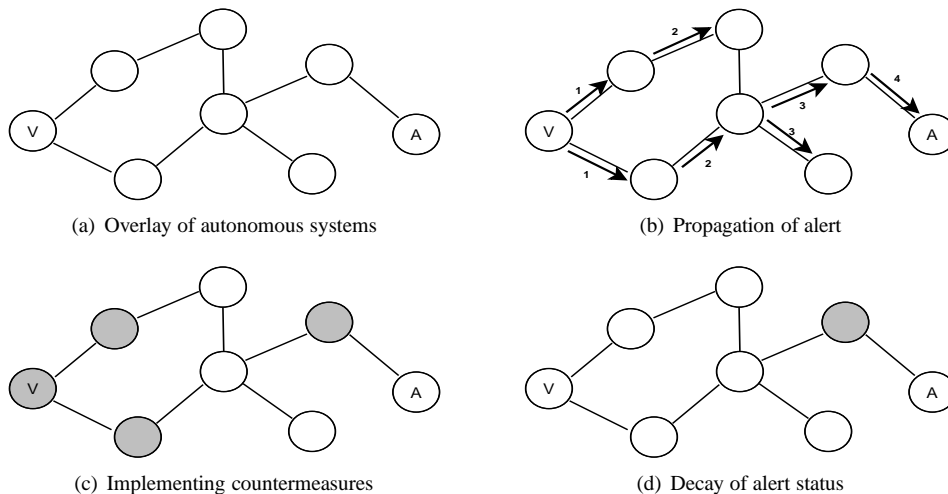
To overcome the problems with the push model, the COVERAGE system implements a pull mechanism in which an organization periodically queries the alert status of a random subset of cooperating nodes to estimate the approximate virulence of a particular worm outbreak [1]. The drawback of this increased resilience to false positives, however, is that the system cannot effectively protect against fast spreading worms because of the delay resulting from the periodicity of alert queries. Most importantly, the countermeasures in the COVERAGE, Nojiri et al., and Kannan et al. systems are implemented to protect only local networks, again requiring significant deployment for effective global worm containment. A discussion of our results in the context of these collaborative systems is given in Section 5.

### 3 Worm Containment and Collaborative Alerting

As work by Wong et al. shows, end-point deployment of worm containment systems is inherently inefficient, and is therefore limited in its ability to provide significant global containment of worm outbreaks [21]. A worm containment system in which the containment occurs within the core of the network, however, should be able to contain extremely virulent worms. Ideally, such containment would occur as close to the infected host as possible, thereby reducing the global impact of the outbreak. Unfortunately, realistic deployment concerns and false alert resilience prevent such a deployment.

Instead, the internetwork-centric containment mechanism described herein aims to provide a scalable implementation which converges toward the optimal blocking scenario as participation in the containment system and breadth of the attack increase. This is achieved by having participating autonomous systems block not only attack traffic destined for local hosts, but also transiting attack traffic destined for other autonomous systems. The discussion of the internetwork-centric containment system is broken into two distinct parts. First, we describe the mechanisms by which containment of worms occurs within the core of the internetwork. Second, we introduce the reputation-based alerting mechanism which provides resilience to false alerts while ensuring fast reaction to worm outbreaks. Throughout the remainder of this paper, we refer to autonomous systems that participate in the containment system as nodes and autonomous systems interchangeably. Also, we emphasize that our containment and alerting systems operate independently of worm detection technologies and can therefore be combined with any number of detection mechanisms. Creation of specific worm detection mechanisms is beyond the scope of this work.

#### 3.1 Internetwork-centric Containment



**Figure 1. Illustration of the internetwork-centric containment mechanism after a single attack instance. (a) A single host within the autonomous system labeled ‘A’ scans a host in the autonomous system labeled ‘V’. (b) An alert is propagated via the shortest path throughout the overlay. (c) The reputation for the attacking host increases, and the shaded autonomous systems implement countermeasures against the host. (d) Reputation decays and only the foremost autonomous system continues blocking due to continued attack traffic from the host; other systems disable their countermeasures.**

When considering the strategy for internetwork worm containment, we must remain cognizant of the unique topology characteristics found within internetwork environments. For instance, the Internet topology has been shown to follow a power-law distribution where a small subset of autonomous systems maintain a large number of connections, while the majority of autonomous systems have few of them [3]. With this in mind, it seems obvious that to best protect the majority of the Internet from worm attack, we would like the autonomous systems with the largest out-degree within the Internet graph to participate in the containment, thereby filtering the worm from all routes that transit them.

Unfortunately, we can not require such participation, so instead we construct a scalable containment system based on an overlay network among the volunteer participants, where adjacencies in the overlay are created through edge contraction between the corresponding autonomous systems within the Internet graph (Figure 1(a)). Edge contraction can be implemented, for example, by examining the AS path field contained in route advertisements for the Border Gateway Protocol (BGP), which provides the shortest route to each of the participating autonomous systems. The containment system messages can then be sent using existing routing infrastructure, just as any other application-layer protocol would be. Several other options exist for the creation of overlay networks, though the specifics of these schemes are beyond the scope of this paper [22, 13, 17].

Upon creation of the overlay network, the alerting mechanism utilizes the overlay topology to propagate the alert from the node that detected the worm attack to its direct neighbors, eventually broadcasting it throughout the network. When an alert is received, the node checks whether it has received an alert for the same attacker from the originating victim within the current so-called management period. If it has, then the alert is dropped, otherwise it is applied and propagated (Figure 1(b)). In essence, the alerts are forced to propagate through the shortest path in the overlay topology. This not only ensures that the same alert is counted only once despite the redundancy that may exist in the overlay topology, but also ensures that a single node cannot perform alert flooding to erroneously cause countermeasures to be implemented at other nodes. The specific implementation of these mechanisms and their respective parameter settings are described in Section 3.2.

Naturally, the node that detected the worm attack would implement filtering countermeasures, such as blocking the offending IP address or their CIDR block immediately, while other nodes in the overlay must reach a certain threshold number of received alerts before implementing their countermeasures (Figure 1(c)). Once a filtering countermeasure is implemented, all traffic that enters the given autonomous system is subject to that filtering policy, including transiting traffic. This ensures that all autonomous systems that would be attacked by routing traffic through the participating system would be protected, as would be the participating autonomous system itself. It is important to note that in order for detection, alert propagation, and containment to occur, the host that is the victim of the attack must be within a participating autonomous system, but the attacking host need not be.

The implementation of countermeasures within the containment system directly corresponds to the breadth of attacks. Therefore, if the attacks are localized to a specific portion of the containment overlay, then the filtering will also be localized due to the properties of the alert propagation. Once attacks from a single host become prevalent enough, we can see that a type of perimeter of blocking would be implemented, which would slowly converge towards the attacker’s autonomous system. This perimeter, of course, is limited by the placement of the participating autonomous systems within the internetwork topology, but converges toward an optimal blocking strategy as participation in the containment system increases.

We further optimize the blocking perimeter by allowing autonomous systems within the overlay to remove their blocking countermeasure when no additional traffic from the offending host is detected; for instance, by using the IP address and port of the malicious traffic as an identifier. The autonomous systems closest to the attacker would continue filtering traffic because it is the so-called first line of defense and is therefore the first participating node to receive the attack traffic. Nodes farther from the attacker will not see the traffic because it is being dropped by the first line of defense, thereby allowing them to remove their filtering countermeasures for that attacker and free resources (Figure 1(d)). This naturally ensures that the global countermeasure strategy for each attacker becomes optimal as the breadth of the attack increases, requiring a minimal number of participating nodes to perform the filtering countermeasures with maximal protection for all hosts.

### 3.2 Reputation-based Alerting

**Table 1. Reputation-based Alerting Notation**

Reputation Variable	Definition
$a$	Attacker
$t$	Current time
$\lambda_i$	Decay period of node $i$
$m_i$	Management period of node $i$
$N_i$	Number of neighbors for node $i$
$rep_{i,y}(x)$	Node $i$ 's reputation value for attacker $x$ at time $y$
$d_i$	Distance from node $i$ to alert originator

Without proper safeguards, a containment system, such as the one proposed here, can easily become a devastating weapon for attackers. Not only should the alerting mechanism prevent the abuse of the containment system for malicious purposes,

but it should also provide fair and fast alerting capabilities that ensure worms are contained in the most efficient way possible. One intuitive notion for implementing alerting mechanisms is the idea of reputation. Without prior experience, humans often use recommendations gained from friends and colleagues to form opinions of various goods, services, and people. Moreover, the use of reputation in social network scenarios has been shown to be an excellent way to quickly propagate information [5].

We leverage a similar concept to provide an alerting mechanism for the internetwork-centric containment system. This alert system allows every participating autonomous system to retain an *independent* reputation value for malicious hosts which can be increased when alerts are received. Hence, this reputation value indicates the level of disrepute for any given host. Once this reputation value increases above a particular threshold at any of the participating autonomous systems, the malicious host’s traffic is blocked at that autonomous system.

The alert system utilizes the overlay network to weight alerts in such a way that the range of alert is limited. This is done for two reasons: (i) to make the overhead of alert messaging scalable as this range is independent of the size of the network, and (ii) to localize the potential attacks against the containment system (e.g., forged alerts). The intuition behind this definition of trust is based on the probability that a given node routes traffic for the local network. It follows that nearby nodes within the internetwork graph necessarily route a larger proportion of the traffic for the local network than more distant nodes. Therefore, if we use BGP routing information to create the overlay topology, as previously described, then the nodes that have close associations within the full internetwork graph will also be nearby in the overlay topology.

The concept of reputation is formalized by a floating point value stored by each node in the overlay and adjusted according to well defined behaviors based on receipt of alerts, and their decay over time. For ease of exposition, we present the notation for our reputation system in Table 1. The system is initialized at each participating node with the reputations for all hosts set to zero. When an alert is received, the value of the attacker’s reputation at the node is increased based upon the distance the alert has traveled in the overlay network, and the number of neighbors for the current node, as shown in (1).

$$rep_{i,t}(a) = rep_{i,t-1}(a) + \frac{1}{(\lfloor \frac{N_i+1}{2} \rfloor d_i) + 1} \quad (1)$$

Clearly, the strength of an alert is multiplicatively reduced based on the distance the alert has traveled before reaching the current node. This limits the effective range of the alert, and creates the localized containment effects that we have described in Section 3.1.

When the distance from the alert originator is zero (i.e. the current node originated the alert), the value of the reputation is increased by one. When the distance from the alert originator is one (i.e. the originator is a direct neighbor), the reputation is increased in such a way that when a majority of the direct neighbors provide alerts, the sum of their alert values will equal one. Thus, we can set the threshold for countermeasure implementation to one, as it allows locally generated alerts to immediately implement countermeasures while requiring the equivalent of a weighted majority vote of remote alerts.

Additionally, we stop the alert propagation once the perceived strength of the alert has reached a level where it will be of no use to other nodes. To accomplish this, a node only forwards an alert to her neighbors if the distance the alert has already traveled is less than the current node’s number of neighbors, excluding the one that sent the alert, which is shown in (2).

$$d \leq (N - 1) \quad (2)$$

The use of both (1) and (2) limits the global effects of any malicious use of the system while allowing for a convergence to optimal global containment from broad worm attacks.

The decay of the reputations stored at each participating node occurs by reducing the reputation at discrete intervals, called *management periods*. These management periods are defined on a per-node basis and indicate the time when the evaluation of the decay in the reputation value actually occurs. Note that this is different than the actual length of the effective decay, known as the  $\lambda$  *period*. Essentially, the  $\lambda$  period is the decay interval, whereas the management period is the interval at which that decay is actually applied and recorded. At each management period, the reputation is reduced multiplicatively by a factor of:

$$\left( \frac{\lfloor \frac{N_i+1}{2} \rfloor}{\lfloor \frac{N_i+1}{2} \rfloor + 1} \right)^{\frac{m_i}{\lambda_i}}$$

When the reputation value is reduced to a value less than one, the countermeasures are stopped. Thus, when the reputation value is exactly one, this decay will remove the equivalent of one direct neighbor alert from the reputation value. While this choice of decay factor is somewhat arbitrary, we believe it provides the benefit of ensuring that the decay is proportional to the connectivity of the node within the network. Therefore, highly connected nodes that would have the most impact on

protecting large portions of the infrastructure require a significant amount of corroboration to enact countermeasures, and the reputation at this node decays very slowly. Conversely, low connectivity nodes that may easily trigger countermeasures have fast reputation decay to prevent any lasting effects of false alerts to local traffic. In addition, this helps ensure that the highly connected nodes bear the brunt of the burden of blocking traffic, which makes the most efficient use of containment resources. The countermeasure threshold can be succinctly derived by combining the required reputation for countermeasure implementation with the decay rate, as provided in (3).

$$reput_t(a) \geq \left(1 + \frac{1}{\lfloor \frac{N+1}{2} \rfloor}\right)^{\lfloor \frac{t}{m} \rfloor \left(\frac{m}{\lambda}\right)} \quad (3)$$

Finally, the reputation-based alerting system ensures that only one alert about an attack by the given attacker from each victim is applied every management period. Not only does this prevent duplicate alerts, but it also has the added benefit of preventing alert flooding. In the case of alert flooding, a single node would send multiple alerts for a single attacker, thereby attempting to erroneously enact a block on it at other nodes in the network. Since only one alert is allowed per management period per victim, there would need to be collusion among malicious nodes to cause erroneous blocking.

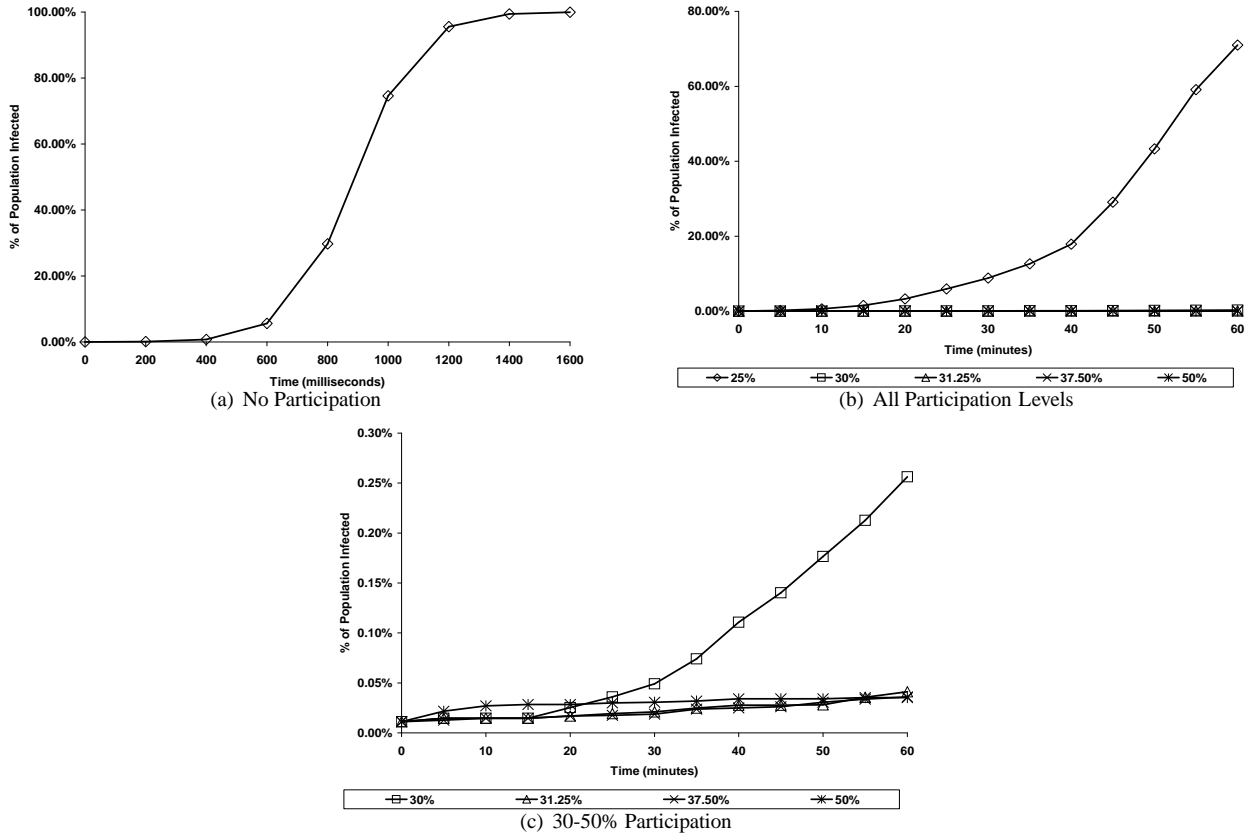
Overall, propagation of alert messages is limited in two ways: (i) each alert has limited range, as to be propagated, an alert message has to be fewer hops from the source than the degree of the node considering propagation (see Eq. 2), (ii) only one alert from a victim of the given attacker is propagated per management period. Hence, the total number of alert messages sent in one management period is a product of the number of attacks times the average propagation area. The latter is independent of the size of the network and defined only by the local topology around each attack victim. Since the management period (10s) is long relative to the regular traffic inter-packet time (typically tens of milliseconds), the alert messages are a very small fraction of the overall traffic.

## 4 Evaluation

To evaluate our worm containment system, we simulate the propagation of an extremely virulent worm within an inter-network of autonomous systems (ASes). We use a custom discrete event simulator to model the propagation of the worm, as well as the reaction of our containment system. The AS topology of our simulation was generated using the BRITE topology generator [7] with parameters derived from the Route Views project [12] and a scaling factor of 1/2 [18]. Specifically, we created a topology of 9,000 ASes with their interconnections distributed according to a power-law distribution [3]. Then, we uniformly distribute an address space of  $2^{31}$  addresses among the ASes, with each address representing a single host.

Our simulated worm has a vulnerable population of 1% of our total address space and a scanning rate of 4,300 scans per second, where scans of the address space occur uniformly at random using the standard C library pseudo-random number generator. In comparison, the worms with historically large vulnerable population, such as Code Red, Blaster, and Slammer, had a vulnerable population reaching up to 0.008% of the total address space [9]. The worm with the fastest scan rate, Slammer, had an effective scanning rate of 4,300 scans per second when considering congestion at access links [8]. The vulnerable population of our simulated worm is uniformly distributed among all ASes in our simulation, and we assume that once a single host in an AS is infected, all hosts in that AS become infected. Though these assumptions on vulnerable population distribution are somewhat unrealistic, they do represent a worst-case scenario for our containment system in which there is a minimum density of alerts per unit of time in any area of the containment overlay topology.

In our simulation, the inter-network-centric containment system is initialized by creating an overlay network of randomly chosen ASes, also using the standard C library pseudo-random number generator. The remaining ASes do not participate in the containment system, but can still become infected. Throughout all tests in our evaluation, we set both the management and  $\lambda$  period to 10 seconds. Since the worm propagates significantly faster than the management and  $\lambda$  periods, their values have no bearing on the efficacy of the containment system. Moreover, unless otherwise specified, the default parameters for the simulation include 50% participation in the containment system, negligible delay in sending alert messages, no dropped alert messages, and a worm detection rate of 100%. Our evaluation will alter each of these parameters independently to ascertain their impact on the performance of the containment system. The worm's infection messages will always propagate with negligible delay and no drops, independent of the parameters for the alert messages. For each parameter setting, we run five simulations over one hour of simulated time. This simulation time has been selected according to the scanning rate, which was set to 4,300 per second. Hence, the simulation time corresponds to time needed for 15 millions of scans, time long enough for the epidemic to flourish or abate. Since the time scale of the worm epidemic dynamics is a function of number of scans rather than time units, 15 million scans can easily translate into days of simulation for slower scanning worms.



**Figure 2. Infection rate of simulated worm with various levels of participation**

We intentionally selected high scanning rate that necessitate computerized deployment of countermeasures that our system enables.

We use the geometric mean of the results over the five simulation runs as an indicator of the average case behavior of the system under those parameter settings, which helps to minimize the stochastic effects of the simulation on the presented results.

**Participation in the Containment System** One of the most important parameters affecting the performance of the containment system is the number of participating ASes. First, we examine the propagation of the worm with no participation in the containment system. As shown in Figure 2(a), our simulated worm is extremely virulent – infecting the entire vulnerable population in just 1.6 seconds. Next, we consider the efficacy of our system with 25, 30, 31.25, 37.5, and 50% participation. Interestingly, the results in Figure 2(b) show that while all levels of participation drastically slow the worm’s propagation, with 25% participation the containment mechanism cannot effectively cutoff the worm’s propagation, and allows 80% of vulnerable hosts to be infected over one hour of simulated time. When we take a closer look at 30, 31.25, 37.5, and 50% participation, shown in Figure 2(c), we see that the containment system achieves containment of the worm in all cases. Also, we see that there is no perceivable gain in performance for participation levels greater than 30% beyond a slight variation in results due to the randomization of participating ASes and the worm’s scanning.

**Effects of Network Conditions** The speed with which our containment system reacts to a worm is based on the propagation of alert messages throughout the overlay network. Therefore, we consider the impact of network conditions, such as dropped packets or delays, on the performance of the containment system. To examine the impact of dropped packets, we drop alerts at any phase of their propagation with 1, 5, and 10% probability. Figure 3(a) shows that there is no discernible difference among the various drop rates beyond stochastic simulation effects. That is, the slight changes in performance are attributable to the random elements of the simulation, and as such we can consider all scenarios to have comparable performance.

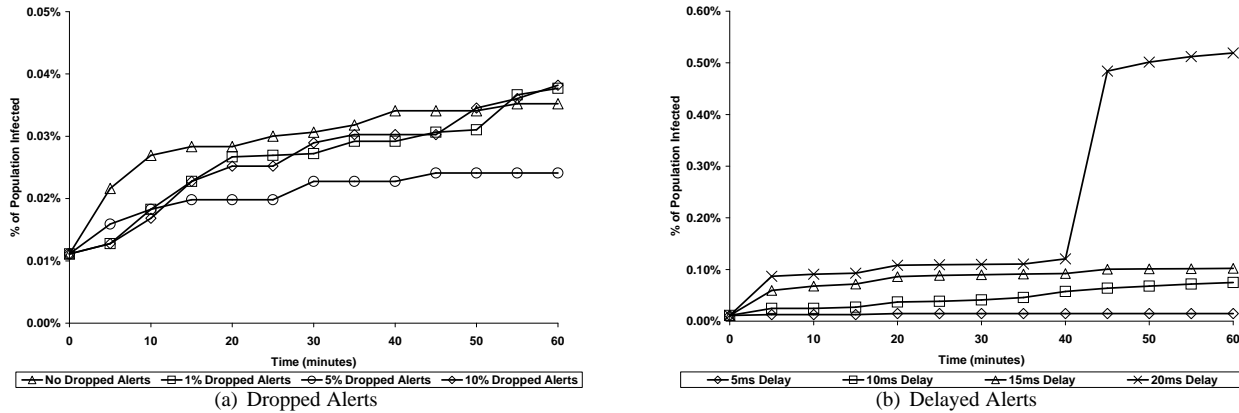


Figure 3. Infection rate of the simulated worm

To examine the impact of delay, we add a delay of 5, 10, 15, and 20 milliseconds to the propagation of alerts at each link in the overlay topology (it should be noted that to achieve these relatively short delays, it may be necessary to use separate (physically or logically) communication channels dedicated to exchanging control messages among containment nodes). Figure 3(b) shows that alert delay has a clear impact on the efficacy of the system, but its effects are somewhat minor – increasing the infected population from 0.05% to 0.6% with a delay of 20 milliseconds over one hour of simulated time.

**Worm Detection and False Alerts** The alerts in our containment system are generated once the victim detects the worm attack using any number of intrusion detection mechanisms. All detection mechanisms have some chance of missing certain attacks, or producing false alerts from legitimate traffic. Furthermore, there is also the distinct possibility that colluding entities could try to generate a large number of false alerts in order to cause a denial of service condition. It is therefore useful to consider the impact that detection rate of the victim’s Intrusion Detection System (IDS) has on the efficacy of the containment system. More importantly, however, we must also examine how counterattacks, such as source spoofed infectious packet flooding, forged alert injection leading to ambient false alerts and colluding ASes affect the operation of the system and its fairness.

First, we examine the effects of detection by having the victim of the worm attack detect the attack and generate an alert with 80, 85, 90 and 95% probability. In Figure 4(a), we see that the detection rate of the victim has no discernible impact on the containment abilities of our system. Again, the differences in performance area result of the simulation’s randomized parameters, and so the results over all detection levels are comparable.

Next, we simulate a false positive in the containment system by forcing 0.5, 1.0, and 2.0% of the participating ASes to generate false alerts about a randomly chosen host at the same time. In effect, this tests our system’s resilience to ambient false alerts independent of assumptions on traffic volume. As expected, Figure 4(b) shows that the number of ASes that have implemented countermeasures as a result of ambient false alerts is linear with the number of nodes providing false alerts. Since the reputation-based alerting system requires corroboration from a number of nodes equivalent to a majority of neighboring ASes, these individual false alarms do not influence the system. In the case where these same ASes collude to produce false alerts for the same host, however, the system causes a large fraction of the participating ASes to implement countermeasures, as seen in Figure 4(b). This result is to be expected since the colluding alerts appear to be identical to the expected behavior of the system in a real worm outbreak.

## 5 Discussion

The results presented in the previous section indicate that our collaborative worm containment system works well in containing even extremely virulent worms. To better understand the significance of these results, we must place them in the context of previous work in collaborative worm containment. Moreover, we also need to analyze the results in terms of their implications for the deployment of the containment system in real internetworks.

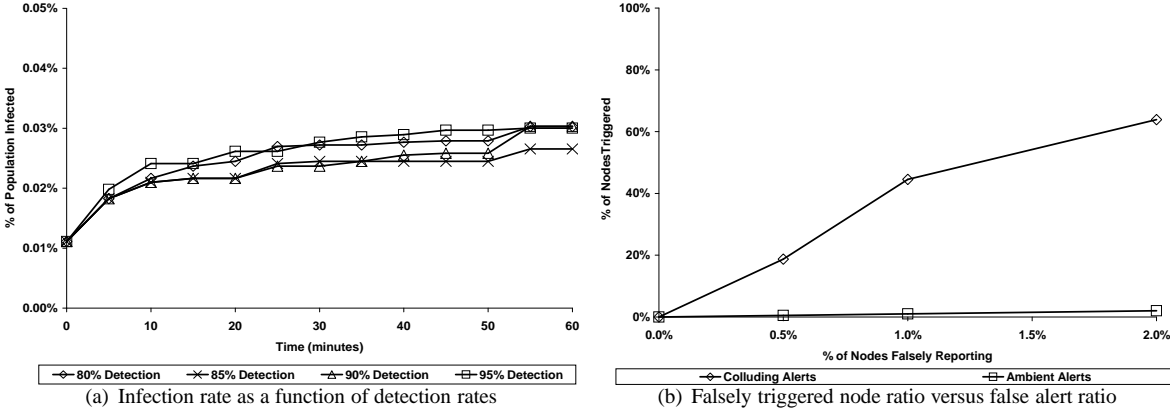


Figure 4. Dynamics of infection detection

## 5.1 Comparison to Previous Work

To provide context for the results presented, we discuss the results of three state-of-the-art collaborative containment systems. The Nojiri et al. and Kannan et al. systems, as described in Section 2, implement a localized containment strategy with a push model for alerting cooperating nodes [11, 4]. The COVERAGE system also implements localized containment, but uses a polling mechanism to share alert information among participants [1]. Note that these systems do not evaluate the impact of colluding participants, and so our comparison is limited to resilience to ambient false alerts and containment performance.

Nojiri et al. create a simulation with 729 networks, all of which participate in the containment system, and 5,832 vulnerable hosts uniformly distributed among all networks [11]. Specific parameters for the scanning rate are not given, but the authors specify that one host is infected per scanning period. The system allows approximately 1,200 hosts (20%) of the vulnerable population to become infected before the worm is stopped. In addition, the authors found that with only 36 (5%) of the participating networks providing ambient false alerts, a total of approximately 550 (75%) networks implemented countermeasures.

Kannan et al. simulate the propagation of a worm with a scanning rate of 20,000 scans per second and a vulnerable population of 0.05% [4]. These vulnerable hosts were uniformly distributed among 10,000 networks where each network has deployed the worm containment mechanism. With their optimal settings, the Kannan et al. system contains the worm to infecting only 0.0005% of the vulnerable population. However, their scheme relies on a static threshold for determining the number of alerts to receive before implementing countermeasures. When this alert setting provides resilience to 1% of the participating nodes providing false alerts, their performance drops by allowing more than 30% of the vulnerable population to become infected.

Finally, COVERAGE creates a network of 2,000 domains with 800,000 hosts uniformly distributed among all domains, all of which have the containment system deployed [1]. Again, COVERAGE does not specify a scanning rate but instead varies the rate of infection attempts *per minute*. With only 0.6 infection attempts per minute, the COVERAGE system is able to reduce the worm to infecting only about 0.005% of the vulnerable population. As the number of infection attempts per minute increases to 2, the COVERAGE system quickly drops its performance allowing nearly 20% of the vulnerable population to be infected. Additionally, when 2% of the participants are providing false alerts, the COVERAGE system triggers countermeasures at 40% of the nodes.

Each of the collaborative containment systems discussed above, therefore, have been evaluated against a worm that is less virulent than the one used in our evaluation in Section 4. Moreover, each of these systems is evaluated with *complete deployment* to all nodes in the simulated environment, whereas our simulation assumes at most a 50% deployment of the system randomly throughout the internetwork. Thus, we evaluated our containment system under weaker assumptions on the environment within which it will be deployed. Even so, our internetwork-centric approach still exceeds the performance of these three previously proposed systems in terms of containment and false alert resilience. For containment performance in the case of 50% deployment, our system allows a maximum of 0.004% of the vulnerable population to be infected while limiting the number of countermeasure triggered by false reports to at most linear function of the number of falsely reporting nodes. A summary of the results for each of these systems, including our internetwork-centric approach, is given in Table 2.

**Table 2. Comparison of results for collaborative containment systems showing their infection rates and associated the rate of countermeasures triggered due to ambient false alerts**

System	% Infected	% False Countermeasures
Nojiri et al. [11]	20%	75%
Kannan et al. [4]	34%	1%
COVERAGE [1]	20%	40%
Internetwork-centric	0.004%	2%

## 5.2 Implications for Real Deployments

At a high level, the internetwork containment system acts as a type of anti-worm that spreads to neighboring autonomous systems (ASes) based upon the topology of the overlay network. Continuing the anti-worm analogy, an ‘infection’ in our containment system occurs when a participating AS receives enough alerts to begin implementing countermeasures. Notice that if our containment system is to protect the vulnerable population, then enough alerts need to be generated in such a way that several participating ASes implement countermeasures *before* more hosts become infected. That is, the spread of our anti-worm must outstrip the growth of the worm. While this is difficult to achieve with false alert resilience, our system is able to do so due to the advantage of internetwork-centric countermeasures.

Since our system can be thought of as an anti-worm, the primary properties affecting its propagation are the level of participation (analogous to vulnerable population) and the amount of delay in propagating alerts (analogous to scanning rate). Of course, our system is implemented in reaction to worm propagation, and so the requirements on these two parameters are closely linked to the virulence of the worm being contained. Thus, more virulent worms require greater participation and faster response than less virulent worms to achieve containment. Unfortunately, a general closed-form analysis of our containment system, similar to epidemiological models for worms, is impossible due to the significant role played by topology in both alerting and countermeasures. That leaves us with the results for our simulated worm, which is highly virulent in comparison to scanning worms observed in the wild [8, 9, 15].

Clearly, our analysis of participation and alert delay in Section 4 underscores their importance in staying ahead of the worm’s propagation. Our evaluation has shown that with as little as 30% participation in the containment system and negligible delay in alert propagation, that the system can completely contain the simulated worm. Similarly, we have also shown that with significant delay and 50% participation, that the system can still effectively contain the worm. It stands to reason that as we increase participation, the tolerance to delay increases. Conversely, as our delay decreases, so can our participation level. The remaining parameters that were tested in our evaluation, such as detection or drop rate, can simply be thought of as probabilistic components of the participation in the system. For instance, with an assumed detection rate of 80%, the effective participation level is, on average, 80% of the actual participation level. A similar argument can be made for dropped alerts, where the effective participation level is a function of the drop rate and the connectivity of each AS in the containment system. Consequently, while the results of our simulation are not generalizable to all worms, we can still use the results to better understand the dynamics of detection, network conditions, and worm virulence on containment.

In addition, we also need to consider the security of the system itself. Since the system enacts proactive countermeasures (e.g., blacklisting), it is of the utmost importance that it cannot be abused by accident or ill intention. Our evaluation has shown that the reputation-based alerting system works as intended by limiting the impact of false alerts on the operation of the containment system. Of course, the AS that generates the false alert would still implement countermeasures, but this would be the case even with stand-alone containment systems.

The more interesting case is that of colluding ASes that intentionally create false alerts. As our analysis has shown, even with 0.5% of the participating ASes colluding, nearly 20% of the participating ASes have enacted countermeasures. While the threshold for enacting countermeasures can be increased, this will have a deleterious impact on containment. Unfortunately, this tradeoff between resilience to collusion and containment will appear in any system using collaborative alerting. A potentially more dangerous attack on our containment system stems from its lack of built-in message integrity and origin authentication. As such, it may be possible for a misbehaving AS to forge messages from other participants in an effort to enact countermeasures on a chosen group of hosts. To avoid these problems, we can adapt existing technologies for securing BGP messages, such as S-BGP [6], to our worm containment system in order to prevent these message forgery attacks.

## 6 Conclusion

All collaborative worm containment systems face a conflict between resilience to false alerts and quick reaction to worm outbreaks. In the past, systems have tended toward either extreme, fast reaction to attacks and poor resilience or good resilience and very slow reaction to worm outbreaks. Furthermore, containment systems, both collaborative and stand-alone, have relied on local countermeasures, which necessitate complete participation in the containment system to protect all vulnerable hosts. In this paper, we have proposed a unique combination of internetwork-centric containment and reputation-based alerting in an effort to provide a more robust containment system suited for use within an untrusted internetwork, like the Internet.

Through simulation studies, we have shown that our containment system can efficiently stop an extremely virulent worm even with relatively little participation. Moreover, we have shown that the system is extremely resilient to ambient false alerts. Finally, we provided a discussion of the parameters affecting the system's performance, and the impact of colluding adversaries on the fairness of the alerting mechanism.

The collaborative containment system proposed here is uniquely suited to providing protection against a wide range of one-to-many attacks, other than scanning worms, that propagate within internetworks. For instance, we might consider spam to be a one-to-many attack where the server sending the spam is the attacker and the recipients are the victims. As with a propagating worm, victims can broadcast an alert and trigger countermeasures that will block the attacker's traffic. To accommodate these other one-to-many attacks, we need only alter the  $\lambda$  and management periods appropriately for their rate of propagation. Overall, the proposed containment system significantly improves on the results of previously proposed collaborative containment systems, and makes a first step toward a much needed autonomic defense for internetworks.

## Acknowledgments

The work of Scott Coull was supported in part by the U.S. Department of Homeland Security Science & Technology Directorate under Contract No. FA8750-08-2-0147. Research of Boleslaw Szymanski is continuing through participation in the International Technology Alliance sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the U.S. Government, the UK Ministry of Defence, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

## References

- [1] K. Anagnostakis, M. Greenwald, S. Ioannidis, A. Keromytis, and D. Li. A Cooperative Immunization System for an Untrusting Internet. In *11<sup>th</sup> International Conference on Networks*, 2003.
- [2] S. Chen and Y. Tang. Slowing Down Internet Worms. In *24<sup>th</sup> International Conference on Distributed Computing Systems (ICDCS)*, pages 312–319, 2004.
- [3] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On Power-Law Relationships of the Internet Topology. In *ACM SIGCOMM*, pages 251–262, 1999.
- [4] J. Kannan, L. Subramanian, I. Stoica, and R. Katz. Analyzing Cooperative Containment of Fast Scanning Worms. In *USENIX SRUTI Workshop*, August 2005.
- [5] H. Kautz, B. Selman, and M. Shah. Referral web: combining social networks and collaborative filtering. *Commun. ACM*, 40(3):63–65, 1997.
- [6] S. Kent, C. Lynn, and K. Seo. Secure Border Gateway Protocol (Secure-BGP). *IEEE Journal on Selected Areas in Communications*, 18(4):582–592, 2000.
- [7] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: An Approach to Universal Topology Generation. In *International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems*, August 2001.
- [8] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the Slammer Worm. *IEEE Security and Privacy*, 1(4):33–39, 2003.
- [9] D. Moore, C. Shannon, and K. Claffy. Code-Red: A Case Study on the Spread and Victims of an Internet Worm. In *2<sup>nd</sup> ACM SIGCOMM Workshop on Internet Measurement*, pages 273–284, 2002.
- [10] D. Moore, C. Shannon, G. M. Voelker, and S. Savage. Internet Quarantine: Requirements for Containing Self-Propagating Code. In *22<sup>nd</sup> Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 1901–1910, 2003.
- [11] D. Nojiri, J. Rowe, and K. Levitt. Cooperative Response Strategies for Large Scale Attack Mitigation. In *DARPA Information Survivability Conference and Exposition*, pages 293–302, April 2005.

- [12] Route Views. <http://www.routeviews.org>.
- [13] A. Rowstron and P. Druschel. Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems. In *IFIP/ACM International Conference on Distributed Systems Platforms*, pages 329–250, November 2001.
- [14] R. Scandariato and J. C. Knight. The Design and Evaluation of a Defense for Internet Worms. In *23<sup>rd</sup> IEEE International Symposium on Reliable Distributed Systems*, pages 164–173, 2004.
- [15] C. Shannon and D. Moore. The spread of the Witty worm. *IEEE Security & Privacy Magazine*, 2(4):46–50, 2004.
- [16] S. Sidiroglou and A. Keryomytis. Countering Network Worms Through Automatic Patch Generation. *IEEE Security and Privacy*, 3(6):41–49, 2005.
- [17] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *ACM SIGCOMM 2001*, August 2001.
- [18] N. Weaver, I. Hamadeh, G. Kesidis, and V. Paxson. Preliminary Results Using Scale-down to Explore Worm Dynamics. In *2<sup>nd</sup> ACM Workshop on Rapid Malcode*, October 2004.
- [19] N. Weaver, S. Staniford, and V. Paxson. Very Fast Containment of Scanning Worms. In *13<sup>th</sup> USENIX Security Symposium*, pages 29–44, 2004.
- [20] M. Williamson. Throttling Viruses: Restricting Propagation to Defeat Malicious Mobile Code. In *18<sup>th</sup> Annual Computer Security Applications Conference*, pages 61–68, 2002.
- [21] C. Wong, C. Wang, D. Song, S. Bielski, and G. Ganger. Dynamic Quarantine of Internet Worms. In *International Conference on Dependable Systems and Networks*, June 2004.
- [22] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz. Tapestry: A Resilient Global-Scale Overlay for Service Deployment. *IEEE Journal on Selected Areas in Communications*, 22(1):41–53, 2004.