Robust Reactions to Potential Day-Zero Worms Through Cooperation and Validation

K. Anagnostakis[1], S. Ioannidis[2], A.D. Keromytis[3], and M.B. Greenwald[4]

[1] Institute for Infocomm Research, 21 Heng Mui Keng Terrace, Singapore 119613
[2] Computer Science Department, Stevens Institute of Technology, Hoboken, NJ 07030, USA
[3] Department of Computer Science, Columbia University, 1214 Amsterdam Ave. New York, NY 10027, USA
[4] Bell Labs, Lucent Technologies, Inc., 600-700 Mountain Ave., Murray Hill, NJ 07974, USA

Abstract. *Cooperative* defensive systems communicate and cooperate in their *response* to worm attacks, but determine the presence of a worm attack solely on local information. *Distributed* worm detection and immunization systems track suspicious behavior at multiple cooperating nodes to determine *whether* a worm attack is in progress. Earlier work has shown that cooperative systems can respond quickly to day-zero worms, while distributed defensive systems allow detectors to be more conservative (i.e. paranoid) about potential attacks because they manage false alarms efficiently.

In this paper we begin a preliminary investigation into the complex tradeoffs in such systems between communication costs, computation overhead, accuracy of the local tests, estimation of viral virulence, and the fraction of the network infected before the attack crests. We evaluate the effectiveness of different system configurations in various simulations. Our experiments show that distributed algorithms are better able to balance effectiveness against viruses with reduced cost in computation and communication when faced with false alarms. Furthermore, cooperative, distributed systems seem more robust against malicious participants in the immunization system than earlier cooperative but non-distributed approaches.

1 Introduction

Increasing innovation among attackers, the increasing penetration of broadband Internet service and persistent vulnerabilities in host software systems have led to new classes of rapid and scalable mechanized attacks on the information infrastructure. Leveling the playing field requires scalable, automated responses to malicious code that can react in the short propagation windows evident with network worms such as Slammer [1]. Traditional approaches have relied on signatures, manual containment and quarantine (*e.g.,* [12]), and while tools are improving, reliance on identifying signatures and other improvements in detection processes is by itself insufficient. What is needed to complete the defensive

S.K. Katsikas et al. (Eds.): ISC 2006, LNCS 4176, pp. 427–442, 2006.

technology portfolio is a scalable, distributed, adaptive response mechanism, based on cooperative behavior amongst a set of responding nodes. Since naïve cooperative behaviors might introduce new risks, including fragility in the face of poor or maliciously-generated information, particular attention must be paid to robustness in the cooperative strategy.

The problem of detecting, quarantining and recovering from day-zero viruses[1] is made easier if local detectors are allowed more room for error. If we err on the side of allowing false alarms, then detectors can be cautious (paranoid!) and conservatively flag anything that looks suspicious, and depend on cooperative corroboration to determine whether the attack is real or not. For this strategy to be effective, though, requires the entire anti-virus system to handle false alarms quickly and cheaply and still respond rapidly to real virus attacks.

Handling individual false alarms is not sufficient, however; by allowing more false alarms we increase the probability that the system will be called upon to manage multiple simultaneous *potential* viral attacks. Simultaneous attacks complicate the anti-virus response because increasing the defense against one virus involves either decreasing the defense against another virus or incurring higher costs (if the system can afford any further anti-virus costs). Simultaneous attacks may occur because of multiple day-zero viruses [10], or because early-stage false-alarms are not yet distinguishable from real virus attacks, or because of the resurgence of old viruses. Old viruses are still potentially virulent, since a measurable fraction of hosts do not upgrade or patch to eliminate security bugs, as the persistence of Code Red and other worms has demonstrated. A good analysis of the persistence of Blaster is given in [4], which shows that tens of thousands of instances of the virus remain active a full year after the initial outbreak. More surprisingly, 73%–85% of infected class-C subnets were not detected as infected during the first Blaster outbreak.

These observations expose several significant problems that must be dealt with. Any node that responds to a potential virus carries a cost: a node has finite resources and therefore can only engage a limited number of viruses at a time. Deciding to counter one virus entails ignoring some other virus. In the absence of cost, the best response to a potential virus attack is to flood the network as rapidly as possible, causing as many cooperating agents to respond at once. The main question is simply whether the response is quick enough to stifle the virus. In the presence of a cost model, however, we still need to respond quickly, but no more quickly than necessary. A false alarm, whether malicious or unintended, can trigger a DoS attack by the response mechanism itself.

In this paper, we investigate tradeoffs in global, distributed response mechanisms that must respond quickly to real viruses and do not over-react to false alarms. These systems should be efficient in terms of bandwidth and global

[1] In this paper we use the term "virus", in an extremely broad manner, to refer to any epidemic-like attack communicated over the network. We use this term, regardless, whether the virus is an active worm that attacks a system even without the unwitting involvement of a legitimate user or a passive virus, that is embedded in a document or email, that requires (unintentional) user assistance to become active.

computation. Moreover, the response mechanism must be robust against malicious agents spreading false information and be able to manage its resources even when many distinct viruses are active at any time. This approach is orthogonal to, and can augment, any proposals for the detection of and recovery from day-zero viruses. It has the added advantage of also performing well in the face of false alarms resulting from malicious behavior or failed detectors.

We focus here on an algorithm called COVERAGE, whose core ideas were originally introduced in [2]. This algorithm takes into consideration shared information about the observed rate of infection for each virus, verifying that new reports are compatible with a node's own empirical observations, and determines (probabilistically) which viruses to respond to. We evaluate its effectiveness through large-scale simulation. We discuss also, although to a lesser extent, tradeoffs in a similar cooperative (but non-distributed) approach, called NRL03, described in [13], which differentiates between slow and fast-spreading viruses.

In our previous work, we determined that our basic cooperative and distributed approach was effective, but only in the sense of measuring the ability of the two approaches to detect and respond to worms of different infection rates, as well as their resistance to malicious nodes that spread misinformation. Here, we offer refined algorithms over the earlier COVERAGE work, and, using a more detailed model we begin to examine the three-way tradeoff between communication costs, computation overhead, and the percentage of the network that gets infected before the reaction mechanism manages to limit the worm propagation. When evaluating alternative approaches in this model, we determine that COVERAGE always has much lower scanning costs; whether in the cases of slowly-propagating worms, fast worms, or in the presence of malicious nodes injecting false alarms. COVERAGE has significantly lower communication cost when dealing with false alarms. Furthermore, the distributed approaches (both COVERAGE and our refined version) can trade off higher communication cost to to detect and react to slow-propagating worms more quickly than the purely cooperative NRL03. Finally, both NRL03 and the COVERAGE variants can use increased communication costs to perform better against fast worms, but the distributed approaches require much higher communication costs than NRL03 — perhaps an unavoidable consequence of their robustness against false alarms.

2 System Model

In this section, we describe our model of how viruses, switches/routers, hosts and our detection mechanism behave.

Modeling Viruses. We use a fairly simple model to describe the behavior of potential attackers (viruses) that we consider in our work. After infecting a node, a virus attempts to infect other nodes; it may attempt to only infect a (small) fixed number of other nodes, or exhibit a greedier behavior. For our purposes, the distinction between the two types is simply in the probability of detection of a probe or attack by a detector. A virus may exhibit high locality of infection

(*i.e.,* probing and attacking nodes based on network-topological criteria, such as "adjacent" IP addresses), or could use a random (or seemingly random) targeting mechanism, *e.g.,* using a large hit-list, or some pseudo-random sequence for picking the next address to attack. We expect that viruses that exhibit high locality are more difficult to detect using an Internet-wide distributed detection mechanism, but easier to do so on a local basis. We completely characterize a virus by the rate at which it attempts to infect other nodes and by the fraction of local attempts it makes. All attacks on susceptible nodes are successful, and in our simulation a virus never attempts to attack a non-existent node. As a result, our simulated viruses are more virulent than equally aggressive viruses in the real world. We make no assumptions about the infection vector: although perhaps the more "interesting" cases are those where the virus is able to automatically subvert a machine or application, our model does not preclude human interaction in the infection process (*e.g.,* mail viruses as attachments).

Furthermore, we only assume that, once detected, there is some detection and/or response "module" associated with each virus — we do not investigate its details: the mechanism may be as simple as a content filter. There is some cost (in terms of CPU, memory, impact on legitimate communications, *etc.*) associated with each of these modules, which requires the prioritization of the various threats (viruses) in terms of allocating resources for detection and response.

Detection of Zero Day Worms. Although our algorithm is orthogonal to and agnostic about the method(s) with which new (zero day) worms are detected, we briefly discuss different techniques and how they may interact with COVERAGE. A zero day worm detector consists of roughly three components. First, we must detect anomalous behavior. The behavior may range from specific activities (e.g. port scans, system/application crashes, incorrect password attempts) to statistical changes (e.g. increased network traffic, slow response time, variation in system call signatures, number of TCP connections in TIME-WAIT). Second, the transmission vector must be identified (finding a set of network packets whose arrival seems to herald the onset of the anomalous behavior). Third, a detectable "signature" of the traffic must be generated so that hosts can scan for, and filter out, the potentially offending traffic. It is important to note that a "signature" in our model is not necessarily simply a pattern of bits to match inside a packet — it can be any profile that detects anomalous behavior, ranging from packet inspection to longer term multi-packet behavior.

Perhaps the most promising approach is that of monitoring the number of packets aimed at the unused portion of an organization's address space, as was suggested in [6]. In that work, it was shown that with as few as 4 such probes, it is possible to infer the existence of a new worm aimed at a previously untargeted service/port. A similar approach is proposed in [19], where sudden changes in the traffic statistics maintained on a per source IP address and per destination port number indicate a high-visibility event, such as a scanning worm. Similar works have proposed measuring the entropy of traffic (*e.g.,* in terms of distinct source IP addresses seen) as an indication of unusual activity. These mechanisms act as early warnings, alerting administrators and perhaps automatically reconfiguring

a firewall to assume a more defensive posture. However, without corroboration with outside sources (*e.g.,* through COVERAGE) they can be manipulated by an attacker to generate false positive reports. It is also worth noting that these mechanisms can only give a rough fingerprint of a new worm's activity, such as the targeted service/port—thus, they can be fairly accurate about the presence of an attack, but inaccurate about mapping specific packets to the attack, as would be the case with a worm targeting a protocol such as HTTP.

A second, more accurate but also more expensive (computationally, as well as in terms of necessary infrastructure) mechanism for detecting worms is through use of properly instrumented honeypots or virtual machines, as is done in [16,9], or through payload analysis [8,17] that can yield a potential worm signature. Finally, anomaly detection techniques, such as those proposed in [5], can indicate the presence of packet payloads that do not conform to the typical contents of packets for a particular service (*e.g.,* binary content containing a buffer overflow payload uploaded to a web server).

These mechanisms identify different points in the zero-day worm detection space, trading off between the likelihood of false positives, the time needed to collect enough evidence before raising an alarm, and the expense of testing whether an alarm should go off.

These observations are taken into consideration by the COVERAGE algorithm to balance the cost of detection (*e.g.,* coordination, scanning as well as collateral damage that may be caused by false alarms) and the ability to respond effectively to virus attacks.

Network Topology. Our simulation topology is dictated by our assumptions about the vulnerabilities and capabilities of network nodes with respect to virus attacks. We assume that, as a general rule, routers/switches are less likely to be infected by a virus, and thus that only hosts are susceptible to infection.

Here, we assume that the only nodes in our system capable of scanning packet sequences for potential viruses are end-hosts or last-hop routers. While considerable advantage can be gained by exploiting the great levels of traffic aggregation seen in routers closer to the network core, it is unlikely that such nodes can actively scan for viruses without significantly affecting their performance.

Thus, our model of the network topology consists entirely of a collection of *subnets* (LANs) containing a number of *hosts*. Each subnet connects to the global network through a single *router*. All routers are connected together in a single cloud where each router can address and forward packets to each other directly. End-hosts can only see their traffic, while routers can inspect all traffic to or from their associated LAN. It is likely that some organizations contain multiple subnets that frequently communicate among themselves. Therefore we collect together several subnets into a *domain*. A domain captures particular communication patterns but has no structural impact on the topology for simulation.

State of Nodes. A node in our environment can be in one of three states with respect to a virus: *susceptible*, *protected*, or *immune*. A susceptible node can be either infected or uninfected. Susceptible nodes will become infected if subjected

to an attack. Protected nodes may be infected or uninfected, but only if the detection module does not have the ability to detect and disinfect an infected machine. A protected node will not become infected as long as the protection mechanism (typically, a module that screens packets or email) is in place. An immune node does not have the vulnerability exploited by the virus.

Operations. A COVERAGE agent can monitor traffic and, for each virus, it can either ignore the virus or perform one or more of the following operations: collect and exchange information about a virus, *scan* for the presence of a virus (actually, scan for the presence of patterns of network traffic used as a "signature" for that virus), or *filter* viruses (by dropping one or more packets that are part of a virus signature). We assume that there is a cost inherent in checking for virus signatures. That is, a node cannot be actively "on the lookout" for an arbitrary number of viruses without adversely affecting its performance. (Some experimental measurements of such real-world limits are given in [3]). Edge-routers are more likely to be constrained by high packet rates, and therefore limited in the amount of scanning they can perform. Hosts can afford to scan for more viruses without interfering with their (lower) packet rate, but, on the other hand, have work other than packet forwarding to perform. In either case there is an upper bound on the number of viruses a node can scan for.

We assume that nodes periodically exchange information about viral infections. Although the per-virus cost of such an exchange is low, we assume that the number of known *plus potential day-zero* viruses exceeds the amount of information that can be reasonably exchanged at any given time. Thus, actively exchanging information about a virus incurs a cost, albeit lower than scanning.

Routers can additionally scan for suspicious behavior on all traffic to or from their LAN (and drop when necessary). We further assume that if a router detects a rampant viral infection for a virus that has an associated disinfectant component, the router can invoke a disinfection operation (perhaps alerting an administrator) on all the nodes in its LAN.

Model of Anti-virus Epidemic. Each node participating in the anti-virus response must make certain decisions: (a) the rate at which it polls other local nodes for virus information, (b) the rate at which it polls other remote nodes, chosen at random, for virus information, (c) whether for each virus to collect information about it, (d), whether to include that information in virus exchange packets, and (e) whether to scan for the virus (collecting the results of those scans as part of the local information for that virus).

3 Cooperative Virus Response

COVERAGE tries to balance the cost of scanning and filtering packets for a specific virus against the benefit of detecting, other, real viruses in several ways. First, COVERAGE models the virulence of viruses and ranks them in virulence order. With probability proportional to their virulence, COVERAGE decides

in rank order whether to actively scan for the virus or not. It stops making decisions, and scans no more viruses, once the scanning schedule consumes the entire scanning budget available. Second, each COVERAGE agent exchanges information about the state of a virus with other cooperating agents in order to construct a model of the virus and determine whether incoming reports are empirically consistent with the observed state of the network. Third, COVERAGE agents determine their polling rate to maximize the probability of seeing enough viruses to confirm the current local estimate of the virus state, while reducing the probability that communication will add no new knowledge to either of the participants. We now describe the algorithm in more detail.

3.1 COVERAGE Algorithm

Agent communication. Each COVERAGE agent polls other agents, selected randomly. Assuming that only a small fraction of the nodes are reporting false information, a randomly selected node is more likely to be trustworthy than a node that actively contacts us — a small number of malicious nodes may try to flood the rest of the network. At each poll, the sender reads the response and updates its local state variables to track the operation of the cooperative response mechanism and the status of the network in terms of observed attacks.

First, it records whether the remote agent is actively scanning. This allows the agent to estimate the fraction of agents in the network that are actively scanning for a particular virus. Second, it updates estimates of possible infections *e.g.,* the fraction of infected nodes for each virus. We distinguish two types of estimates: direct and remote. Direct estimates are updated based on whether each remote agent has directly observed an attack (either to itself or, if a router, to a node in its LAN). Remote estimates are updated based on the fraction of infected nodes as estimated by the remote agent (the "direct" estimates of the remote agent). Direct measurements performed by the local node are absolutely trustworthy — there is no issue of false positives. The direct measurements of agents that we poll (which become our remote estimates) are next in trustworthiness. Remote estimates of agents who we poll are more suspect, and information reported by agents who contact us are the most suspicious of all. However, we can validate any information reported to us — if someone reports that a particular virus is attacking 25% of the Internet at the moment, then if we poll 20 agents at random, then with 80% probability we would expect to find that between 3 and 7 of those agents had directly seen an attack in the last measurement interval. Values outside that range would cast doubt on the remote estimate.

Finally, in this paper we ignore the details of how COVERAGE nodes authenticate themselves to each other. However, we note that even strong authentication is not sufficient for our system. If a COVERAGE agent is taken over by a malicious attacker, then the attacker can (presumably) still authenticate itself and discover which nodes are *not* scanning for a particular virus, and use that information when choosing targets. To defend against such a vulnerability in COVERAGE , we propose (but have not yet implemented or experimented with) a simple defense. When polling, the identity of the target agent is not important — just the fact that we

chose it randomly, and it did not choose us. And, while we are interested in the statistics of the sample as a whole, we need not link a particular set of direct measurements to a particular IP address. Consequently, each agent stores a randomly selected response from the last measurement interval (the local measurements are one of the candidates that may be selected), and returns that random selection in response to any COVERAGE poll, for the direct measurements and scan list only. (The cumulative counters are still stored and reported accurately). The poller still receives an accurate response — just perhaps from a different IP address than the one it polled, and perhaps slightly older than expected. This adds a level of indirection to the polling process.

Periodic updates. At regular intervals each COVERAGE agent updates its state based on the information received since the last update. To track the progress of the infection each COVERAGE agent maintains a smoothed history for each type of estimate (direct and remote), each as exponentially decaying averages with varying time constants, to approximate recent infection rate, past rate, and background rate.

Using these estimates, an agent can compute the fraction of nodes believed to be infected as well as the growth of the infection, assuming exponential growth[2].

If we assume that each infected node infects α nodes at each timestep, and that a fraction $p^\star$ of all nodes are infected at some start time t_0, then at time t we expect the fraction of infected nodes to be $p^\star(1+\alpha)^{t-t_0}$. Consequently, if our direct samples at times t_0, t_1, and t_2 report a fraction $p_d[t]$ nodes are infected[3], we can estimate $p_d^\star$ and α_d for future growth by fitting

$$\begin{aligned} p_d[t_0] &= p_d^\star \\ p_d[t_1] &= p_d^\star(1+\alpha_d)^{(t_1-t_0)} \\ p_d[t_2] &= p_d^\star(1+\alpha_d)^{(t_2-t_0)} \end{aligned}$$

Given estimates of $p_d^\star$ and α_d, we can calculate the *virulence*, v_d, of a virus as the estimated number of timesteps needed by the virus to infect the entire network.

Note that we independently calculate virulence for global and local growth, in order to identify attacks that are non-uniformly distributed throughout the network. Using the same method as above the agent also computes α_r, $p_r^\star$ and v_r based on the remote estimates.

Scanning/filtering. Given the estimates an agent can decide whether it needs to scan for a given virus. There is a basic, low level of scanning for every virus.

[2] We assume all growth is exponential for the purpose of deciding whether to trigger a reaction. We believe that linear growth worms can be detected by humans, and need not be countered by an automatic, distributed, algorithm. If our assumption is incorrect and growth is, in practice, sub-exponential then we recover naturally because we observe a decrease in α and gradually back-off as the predicted "virulence" of the virus drops.

[3] In fact, we use the smoothed averages rather than instantaneous samples, and the details of the actual calculation are a bit more complicated, but are not relevant to the main point of this paper.

When a virus becomes active the scanning rate may increase. In the general case, the agent can sort viruses in order of their virulence v_d and decide whether to scan for each virus, in turn, stopping when the scanning budget is filled. (In our simulation, we only scan viruses whose v_d is below *threshold*.)

To maintain a basic, low level of scanning for every virus, every agent measures the fraction $f_{scanning}$ of nodes in the network that are actively scanning for a given virus based on information exchanged with other nodes. If this fraction is below a threshold f_{target} (around 2-5%) and the node has enough resources for scanning, it activates with probability $f_{target} - f_{scanning}$, and disables scanning in a similar way if too many nodes seem to be active. To avoid turning "blind" to certain worms because of a fraction of malicious nodes falsely reporting that they are actively scanning, nodes need to aim for $f_{target} + f_{malicious}$, where $f_{malicious}$ is the maximum tolerable fraction of malicious nodes. Although this increases scanning cost, nodes can trade-off this cost for higher communication costs.

An inactive agent, A, may also start scanning seemingly low-virulence viruses, if *enough other* agents claim the virus is virulent, and A finds that the fraction of scanning nodes is too low to detect virus activity in a single timestep at the current polling rate. The test is whether n (simply the fraction of agents that were polled and found to be scanning in the last interval) is less than twice the estimated fraction of infected hosts (*e.g.,* if $n < 2p_r^\star$). Similarly, if the agent is active but $n > p_r^\star$ then it decides to stop. The agent also stops scanning if α_r approaches 0. This ensures that the fraction of scanning agents is bounded if there is insignificant progress for a given infection or if the infection is small compared to the number of actively scanning agents. Such heuristics are essential for controlling the behavior of the algorithm, keeping the response mechanism "ahead" of the virus but also limiting the damage and cost when malicious agents spread false information.

A small number of agents need to be watching for each dormant virus. The number of active scanners monitoring a virus may be more than warranted by the level of virus activity. An agent detecting this will stop monitoring the virus. If the agent finds that it now has ample room within its scanning budget to consider another virus, it chooses another virus to monitor uniformly at random from the (large) virus database. The agent may choose a virus that almost no one else is scanning for — in which case it will stay on the scanning list for a long time, and be inspected by the agent as long as there are not too many virulent virii. If the new virus is dormant, but enough people are already looking at it, then the agent will drop it, and randomly choose another.

Polling rate. An agent communicates with agents within the same domain at a constant, high rate, as the cost of intra-domain communication is assumed to be very small. Inter-domain communication is generally more expensive; agents therefore need to adapt the rate of polling remote agents, avoiding excessive communication unless necessary for countering an attack. When there is no virus activity, agents poll at a pre-configured minimum rate (at least an order of magnitude lower than the rate for intra-domain communication). An agent periodically adapts the remote polling rate if v_r is less than a given threshold.

The new rate is set so that the agent polls $1/(p_r^\star)^2$ remote agents in each update interval, unless this rate exceeds a pre-configured maximum rate. This is used to increase the polling rate when the remote estimate indicates that an attack is imminent (but not yet reflected in the direct estimate). If the more recent direct estimate $p_d[n]$ is non-zero, then the polling rate is increased so that at least a few samples can be collected in each update interval. Finally, if the estimated virus population $p_r^\star$ is small and the estimated virus growth rate is close to zero, the agent throttles back its remote polling rate to the minimum rate.

These adjustments are always performed on the polling side. We avoid changing the state or behavior of the polled agent to reduce the risks associated with malicious agents. Otherwise, they could spread misinformation and raise false alarms more effectively by increasing their own communication rate.

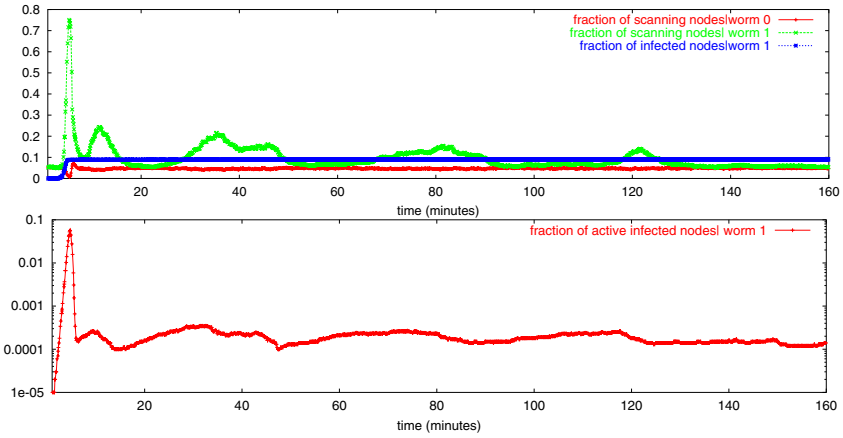

Fig. 1. Fractions of infected hosts and scanning nodes over time (top), and fraction of actively infected nodes (bottom)

3.2 COVERAGE Behavior

To give a rough sense of how the COVERAGE algorithms described above behave, Figure 1 displays a single example run of the COVERAGE algorithm against a single simulated virus called "worm 1". We show the activity of the virus (the number of nodes that were ever infected in their lifetime) in (a), and the currently infected nodes in (b)), as well as the response of COVERAGE (both the number of agents scanning for "worm 1", as well as the number of agents scanning for a dormant virus "worm 0"). (Section 4 describes how we approximate a heavy load on the COVERAGE agents by using a simulation parameter `threshold` — each agent is too busy to consider any virii unless they are likely to take over the entire network within `threshold` measurement intervals.)

One can see the initial stage of the infection and the response of the algorithm: the virus manages to infect roughly 10% of the hosts; cooperation between COVERAGE agents results in a rapid activation of filtering on roughly

75% of the network effectively eliminating the virus. Soon after stopping the attack, the COVERAGE agents on uninfected parts of the network deactivate scanning/filtering. However, as shown in Figure 1(b), a small number of hosts remains infected and undiscovered, resulting in another three episodes where COVERAGE agents are activated (each episode with a smaller fraction of agents activated) to defend against a secondary outbreak. Although a tiny fraction of infected nodes remains undiscovered, it does not cause any further harm and COVERAGE gives users time to patch up their systems. The scanning for dormant "worm 0" continues, except during the most virulent part of the outbreak, where the number dips as resources are marshaled to defend against "worm 1".

4 Simulation Results

To simplify the analysis of COVERAGE and to meaningfully include the non-adaptive NRL03, we restrict the simulation to a single virus. We model the impact of multiple active viruses by assuming that each node is busy handling other viruses. To represent the load imposed by other viruses, we specify a threshold under which a virus will not have high enough priority to be scheduled in the scanning budget. If many viruses are active then the threshold will be a small number, such as 5 (recall that the virulence is a measure of how many measurement intervals it will take before the virus has covered the *entire* Internet). Unless the current virus is poised to conquer the entire net at its current rate of growth from its current coverage within `threshold` intervals, it will not have high enough priority to be scheduled in the scanning budget. We only consider cases where the net is already under heavy attack by other viruses, setting threshold equal to 5 and 20.

To better understand the performance of COVERAGE, we limit our simulation to a simple, relatively small network of 100,000 edge-routers, each connected to 8 hosts, with 50 edge-routers in each of 2,000 domains. We also consider the performance of our version of COVERAGE in relation to NRL03 [13], another cooperative algorithm, which makes different tradeoffs than COVERAGE. NRL03 uses cooperative peer-to-peer strategies to respond to large scale Internet worm attacks. The model involves a number of *friend* nodes, which work together by exchanging information to warn of suspicious worm-like network behavior. As in COVERAGE, a small fraction of nodes is assumed to be scanning for a given virus. When the virus is detected, the node broadcasts the alert to its friends. When a node receives such an alert, it increments an alert counter, and propagates the alert to its set of friends when this counter reaches a threshold.

For the COVERAGE algorithm, we set the local-domain polling interval to 1.8 seconds , the maximum and minimum remote polling intervals to 6 seconds and 1.8 seconds respectively. For both algorithms we assume that 4% of the edge-routers are permanently scanning for the virus.

Our analysis uses three metrics. First, we model the success of the attack by integrating the number of infected nodes over time. This is only relevant in

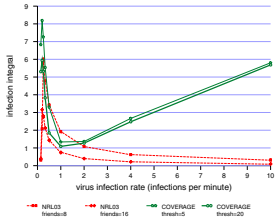

Fig. 2. Integral of infected hosts over time *vs.* virus infection rate

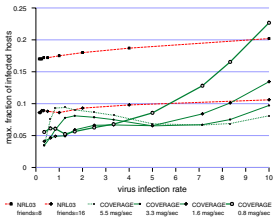

Fig. 3. Maximum fraction of infected hosts vs. virus infection when under attack

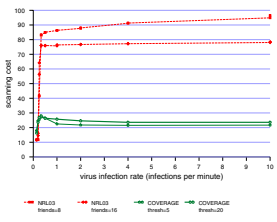

Fig. 4. Scanning activity *vs.* virus infection rate when under attack

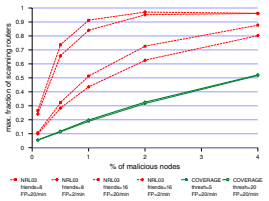

Fig. 5. Maximum fraction of false scanning routers with malicious nodes and false alarms

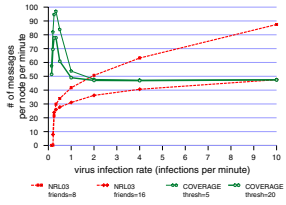

Fig. 6. Communication cost *vs.* virus infection rate when under attack

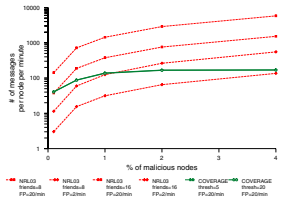

Fig. 7. The impact of malicious routers and false alarms on communication cost

the case of a real virus attack. Second, we consider the number of edge-routers actively scanning/filtering this virus. This is a measure of the computational overhead of the response mechanism. Our third metric, the total number of messages sent, measures the communication cost.

We measure the progress of infections of differing virulence and the success of the response mechanism as the integral over time of the fraction of infected nodes. The results for COVERAGE and NRL03 with different parameters are shown in Figure 2. (It may seem counter-intuitive that the more virulent viruses cover less of the network; however, recall that we are integrating over time and that the faster worms, while they spread faster are also detected and disinfected faster). COVERAGE reacts more slowly than NRL03 for fast worms — we model COVERAGE dealing with other viral outbreaks, but let NRL03 assume that this is the only virus in the Internet. Consequently, the virus takes a larger initial toehold in the network under COVERAGE, and is active slightly longer before being cleaned up. Because of this toehold, COVERAGE performs relatively worse than NRL for fast worms. On the other hand, it detects the slow worms before NRL03, and therefore does better. The slow response by COVERAGE in the case of fast worms has been a deliberate design choice in an attempt to make the algorithm robust against false information from malicious nodes. Figure 3 plots the high water mark of viral attacks as a function of the virus infection rate for different communication settings in COVERAGE. We can see that a moderate increase in communication rates in COVERAGE allows it to

stop the virus with a lower high water mark than NRL, but at the expense of more communication. [4]

Figures 4 and 5 show the fraction of nodes scanning for a virus as a function of the virulence of the virus and the fraction of malicious (or faulty) nodes. The figures for COVERAGE are more pessimistic than for NRL, because in NRL *every* edge-router is scanning for the virus (obviously impractical for a large set of viruses), and the graphs report only those nodes that are actively *filtering* the virus. The COVERAGE plots report the fraction of nodes that even scan for the virus at all. A smaller number (unreported here) are filtering for the virus. Nevertheless, a pessimistic ("worst-case") results for COVERAGE have far lower scanning costs than optimistic ("best-case") results for NRL03. COVERAGE manages to control the virus with a much smaller set of scanning nodes, and it similarly detects false alarms with fewer nodes triggered to scan or filter.

Figures 6 and 7 demonstrate that the communication costs for COVERAGE in the face of false alarms is much lower than for NRL03 — understandably because COVERAGE identifies the false alarms correctly. In the case of slow-growth worms, COVERAGE requires significantly more communication to convince cooperating peers that a virus attack *is* underway. However, this extra cost conveys a benefit: COVERAGE detects slow-growth worms long before NRL03 is able to. For fast worms, communication costs are generally comparable — NRL requires considerably more communication when Friends = 8, but it should be noted that NRL03 controls the infection more rapidly than COVERAGE in these cases. For COVERAGE to control fast worms as effectively as NRL03 would require even higher communication costs.

The impact of false alarms on detection performance (because nodes get confusing reports from malicious nodes about another virus) is illustrated more clearly in Figure 8. We see that the fraction of nodes that are left unprotected by COVERAGE grows almost linearly with the number of malicious nodes – roughly double the number of nodes are infected when 4% of the nodes are malicious compared to a system without any malicious nodes.

5 Related Work

Reference [15] shows that a distributed worm monitor can detect non-uniform scanning worms two to four times as fast as a centralized telescope, and that knowledge of the vulnerability density of the population can further improve detection time.

In [11], the authors coordinate the sharing of IDS alerts for detecting worm attacks and port scanning across administrative domains. Porras *et al.* [14] argue that hybrid defenses using complementary techniques (in their case, connection

[4] The communication costs for COVERAGE scale with the virulence and number of active viruses, and are thus more scalable than NRL — still, we are investigating ways of conveying the necessary polling information more efficiently during quiet periods.

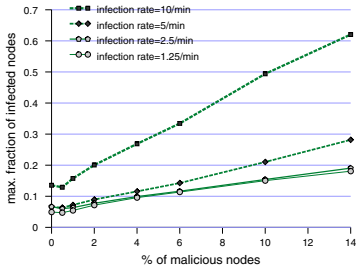

Fig. 8. Impact of false alarms and malicious nodes on detection performance

throttling at the domain gateway and a peer-based coordination mechanism), can be much more effective against worms.

Reference [18] proposes the use of "predator" viruses that spread in much the same way malicious viruses do but try to eliminate their designated "victim" viruses. The authors show that predators can be made to perform their tasks without flooding the network and consuming all available resources. However, designers of predators would have to find their own exploits (or safeguard exploits for future use), which is not an attractive proposition. Furthermore, many recent worms have been closing the hole they exploited, after infecting a machine.

DOMINO [20] is an overlay system for cooperative intrusion detection. The system is organized in two layers, with a small core of trusted nodes and a larger collection of nodes connected to the core. The experimental analysis demonstrates that a coordinated approach has the potential of providing early warning for large-scale attacks while reducing potential false alarms. Reference [21] describes an architecture and models for an early warning system, where the participating nodes/routers propagate alarm reports towards a centralized site for analysis. The question of how to respond to alerts is not addressed, and, similar to DOMINO, the use of a centralized collection and analysis facility is weak against worms attacking the early warning infrastructure.

The earliest work on cooperative response mechanisms is that of Nojiri *et al.* [13]. They present a cooperative response algorithm where edge-routers share attack reports a small set of other edge-routers. Edge-routers update their alert level based on the shared attack reports and decide whether to enable traffic filtering and blocking for a particular attack. Analysis by Kannan et al [7] has shown that cooperative response algorithms can improve containment, even when a minority of firewalls cooperate. That work, however promising, does not directly relate to our work. They are more concerned with a single fast virus — the analysis focuses on a single virus (consequently underplaying the cost of over-aggressive response), has a weaker model of "malicious" firewalls (malicious firewalls merely stay silent, but do not mislead through false alarms), and does not explore the benefits of allowing more lattitude in generating false alarms.

6 Conclusions and Future Plans

We have described an algorithm, named COVERAGE, that allows cooperating agents to share information about the spread of malicious virus in the Internet and use this information for controlling the behavior of detection and filtering resources. The algorithm operates without fully trusting such information, so as to limit the damage of false alarms injected by malicious nodes. Our solution is based on the idea of carefully sampling of global state to validate claims made by individual participants. Simulation results confirm that this method is effective in limiting the damage of virus attacks, and that it is robust against attacks by malicious participants. When compared against a similar approach, the NRL03 algorithm [13], COVERAGE exhibits a lower cost in terms of scanning for worms due to its resource-aware approach. Furthermore, it has a lower communication cost in the presence of false alarms and fast worms, and can detect and react to slow-propagating worms better. However, it does not react as quickly to fast worms, and the price it pays for reacting to slow worms more quickly is higher communication overhead than NRL03 for slow worms.

Our plans for future work include reducing the communication cost of polling without measurably reducing the effectiveness of the mechanism, and examining in detail the case of multiple, simultaneously active viruses. We believe that we can tune the communication rate to adapt to the virulence of a virus, allowing COVERAGE to react to fast worms quickly (at the cost of increased communication overhead for very virulent worms). Due to space considerations, this paper simply assumes that many COVERAGE nodes are occupied by higher-virulence viruses, and that we must reserve processing cycles to deal with lower virulence viruses, too. Much work remains to be done in improving the actual choices each node makes of which set of viruses to monitor.

Acknowledgements. This work was supported in part by the National Science Foundation under grants ITR CNS-0426623, DUE-0417085 and CCR-0331584.

References

1. Cert Advisory CA-2003-04: MS-SQL Server Worm. `http://www.cert.org/advisories/CA-2003-04.html`, January 2003.
2. K. G. Anagnostakis, M. B. Greenwald, S. Ioannidis, A. D. Keromytis, and D. Li. A Cooperative Immunization System for an Untrusting Internet. In *Proceedings of the* 11^{th} *IEEE Internation Conference on Networking (ICON)*, pages 403–408, September/October 2003.
3. K. G. Anagnostakis, M. B. Greenwald, S. Ioannidis, and S. Miltchev. Open Packet Monitoring on FLAME: Safety, Performance and Applications. In *Proceedings of the* 4^{th} *International Working Conference on Active Networks*, December 2002.
4. M. Bailey, E. Cooke, F. Jahanian, D. Watson, and J. Nazario. The Blaster Worm: Then and Now. *IEEE Security & Privacy*, 3(4):26–31, July/August 2005.
5. M. Bhattacharyya, M. G. Schultz, E. Eskin, S. Hershkop, and S. J. Stolfo. MET: An Experimental System for Malicious Email Tracking. In *Proceedings of the New Security Paradigms Workshop (NSPW)*, pages 1–12, September 2002.

6. J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan. Fast Portscan Detection Using Sequential Hypothesis Testing. In *Proceedins of the IEEE Symposium on Security and Privacy*, May 2004.
7. J. Kannan, L. Subramanian, I. Stoica, and R. H. Katz. Analyzing Cooperative Containment of Fast Scanning Worms. In *Proceedings of Steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI)*, pages 17–23, July 2005.
8. H. Kim and B. Karp. Autograph: Toward Automated, Distributed Worm Signature Detection. In *Proceedings of the* 13^{th} *USENIX Security Symposium*, pages 271–286, August 2004.
9. J. G. Levine, J. B. Grizzard, and H. L. Owen. Using Honeynets to Protect Large Enterprise Networks. *IEEE Security & Privacy*, 2(6):73–75, Nov/Dec 2004.
10. E. Levy. Approaching Zero. *IEEE Security & Privacy*, 2(4):65–66, July/August 2004.
11. M. Locasto, J. Parekh, S. Stolfo, A. Keromytis, T. Malkin, and V. Misra. Collaborative Distributed Intrusion Detection. Technical Report CUCS-012-04, Columbia University Department of Computer Science, 2004.
12. D. Moore, C. Shannon, G. Voelker, and S. Savage. Internet Quarantine: Requirements for Containing Self-Propagating Code. In *Proceedings of* 22^{nd} *Annual Joint Conference of IEEE Computer and Communication societies (INFOCOM)*, April 2003.
13. D. Nojiri, J. Rowe, and K. Levitt. Cooperative response strategies for large scale attack mitigation. In *Proceedings of the* 3^{rd} *DARPA Information Survivability Conference and Exposition*, April 2003.
14. P. Porras, L. Briesemeister, K. Levitt, J. Rowe, and Y.-C. A. Ting. A Hybrid Quarantine Defense. In *Proceedings of the ACM Workshop on Rapid Malcode (WORM)*, pages 73–82, October 2004.
15. M. A. Rajab, F. Monrose, and A. Terzis. On the Effectiveness of Distributed Worm Monitoring. In *Proceedings of the* 14^{th} *USENIX Security Symposium*, pages 225–237, August 2005.
16. S. Sidiroglou and A. D. Keromytis. A Network Worm Vaccine Architecture. In *Proceedings of the IEEE Workshop on Enterprise Technologies: Infrastructure for Collaborative Enterprises (WETICE), Workshop on Enterprise Security*, pages 220–225, June 2003.
17. S. Singh, C. Estan, G. Varghese, and S. Savage. Automated worm fingerprinting. In *Proceedings of the* 6^{th} *Symposium on Operating Systems Design & Implementation (OSDI)*, December 2004.
18. H. Toyoizumi and A. Kara. Predators: Good Will Mobile Codes Combat against Computer Viruses. In *Proceedings of the New Security Paradigms Workshop (NSPW)*, pages 13–21, September 2002.
19. J. Wu, S. Vangala, L. Gao, and K. Kwiat. An Effective Architecture and Algorithm for Detecting Worms with Various Scan Techniques. In *Proceedings of the Network and Distributed System Security (NDSS) Symposium*, pages 143–156, February 2004.
20. V. Yegneswaran, P. Barford, and S. Jha. Global Intrusion Detection in the DOMINO Overlay System. In *Proceedings of NDSS*, February 2004.
21. C. C. Zou, L. Gao, W. Gong, and D. Towsley. Monitoring and Early Warning for Internet Worms. In *Proceedings of the* 10^{th} *ACM International Conference on Computer and Communications Security (CCS)*, pages 190–199, October 2003.