Optimal Cost, Collaborative and Distributed Response to Zero-Day Worms - A Control Theoretic Approach

Senthilkumar G. Cheetancheri[1], John-Mark Agosta[2], Karl N. Levitt[1], Jeff Rowe[1], and Felix Wu[1]

[1] Security Lab, Dept. of Computer Science, Univ. of California, One Shields Ave., Davis, CA - 95616, USA
[2] Intel Research. 2200, Mission College Blvd., Santa Clara, CA - 95052, USA

Abstract. Collaborative environments present an happy hunting ground for worms due to inherent trust present amongst the peers. We present a novel control-theoretic approach to respond to zero-day worms in a signature independent fashion in a collaborative environment. A federation of collaborating peers share information about anomalies to estimate the presence of a worm and each one of them independently chooses the most cost-optimal response from a given set of responses. This technique is designed to work when the presence of a worm is uncertain. It is unique in that the response is dynamic and self-regulating based on the current environment conditions. Distributed Sequential Hypothesis Testing is used to estimate the extent of worm infection in the environment. Response is formulated as a Dynamic Programming problem with imperfect state information. We present a solution and evaluate it in the presence of an Internet worm attack for various costs of infections and response.

Keywords: Security, Collaboration, Dynamic Programming, Control Theory.

1 Introduction

Computer worms are a serious problem. Particularly in a collaborative environment, where the perimeter is quite secure but there is some amount of trust and implicit security within the environment. Once, a worm breaks the perimeter defense, it essentially has a free run within the collaborative environment. An enterprise environment is a typical example of a network with this 'crunchy on the outside – chewy on the inside' characteristic. In this paper, we try to leverage the collaboration to collectively defend against such worm attacks. Dealing with known worms is a solved problem – signatures to be used by Intrusion Prevention Systems(IPSs) are developed to prevent further infections, and patches are developed to fix vulnerabilities exploited by these worms. Dealing with unknown worms – worms that exploit zero-day vulnerabilities or vulnerabilities for which patches have either not been generated or not applied yet – is still a research question. Several ingenious proposals to detect them automatically exist. Many

sophisticated counter measures such as automatic signature generation and distribution [10,13,14,16] and automatic patch generation to fix vulnerabilities [15] have also been developed.

Often times, even if automated, there is not much time to either generate or distribute signatures or patches. Other times, system administrators are skeptical about applying patches. During instances when response based on the above mentioned techniques are not feasible, the only option left is to either completely shut-down the vulnerable service or run it risking infection. It is usually preferred to shut-down the service briefly until a mitigating response is engineered manually.

However, making a decision becomes hard when one is not certain if there is really a worm, and if the service being offered is vulnerable to it. It is not desirable to shut-down a service only to realize later that such an action was unwarranted because there is no worm. However, suspending the service in an attempt to prevent infection is not considered bad. Intuitively, it is desired to suspend the service briefly until it is clear whether there is an attack or not. Balancing the consequences of providing the service risking infection against that of not providing the service is of the essence.

This paper captures this intuition and devises an algorithm using Dynamic Programming(DP) techniques to minimize the overall cost of response to worms. Cost is defined as some mathematical expression of an undesirable outcome.

These algorithms use information about anomalous events that are potentially due to a worm from other co-operating peers to choose optimal response actions for local application. Such response can be later rolled-back in response to changes to the environment such as a curtailed worm. Since peers decide to implement response independently, the response is completely decentralized.

We surprisingly found that in certain scenarios, leaving oneself open to infection by the worm might be the least expensive option. We also show that these algorithms do not need large amounts of information to make decisions. One of the key achievements here is that we use weak Intrusion Detection Systems(IDSs) as sensors which have high false positive rates. By corroborating alerts raised by them with other collaborating sensors, we are able to minimize the false positives and achieve better fidelity in detecting worms.

2 Dynamic Programming

This section provides a brief introduction to the theory behind Dynamic Programming [4]. DP as applied to the current problem balances the low costs presently associated with operating a system against the undesirability of high future costs. The basic model of such a system is dynamic and discrete with an associated cost that is additive over time. The evolution of such a system can be described as:

$$x_{k+1} = f_k(x_k, u_k, w_k), \quad k = 0, 1, \ldots, N-1 \ , \tag{1}$$

where k indexes discrete time, x_k is the state of the system and summarizes past information that is relevant for future optimization, u_k is the control or

decision variable to be selected at time k, w_k is a random parameter, also called disturbance or noise depending on the context, N is the horizon or the number of times control is applied and f_k is the mechanism by which the state is updated. The cost incurred at time k is denoted by $g_k(x_k, u_k, w_k)$ which is a random function because it depends on w_k. The goal is to minimize the total *expected cost*

$$J_\pi(x_0) = \underset{w_k}{E}\left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k) \right\} \ .$$

This is achieved by finding a sequence of functions called the *policy* or *control law*, $\pi = \{\mu_0, \ldots, \mu_{N-1}\}$, where each $\mu_k(x_k) \rightarrow u_k$ when applied to the system takes it from state x_k to x_{k+1} and minimizes the total *expected cost*. In general, for a given π, we use $J_k(x_k)$ to denote the *cost-to-go* from state x_k at time k to the final state at time N.

Dynamic Programming Algorithm: The optimal total cost is given by $J_0(x_0)$ in the last step of the following algorithm which proceeds backwards in time from period $N-1$ to period 0:

$$J_N(x_N) = g_N(x_N), \tag{2}$$

$$J_k(x_k) = \min_{u_k} \underset{w_k}{E}\Big\{ g_k(x_k, u_k, w_k) + J_{k+1}(x_{k+1}) \Big\}, \quad k = 0,\ 1,\ \ldots,\ N-1 \ . \tag{3}$$

2.1 Imperfect Information Problems

DP problems as described above have perfect information about the state of the system, x_k. Often, x_k cannot be determined accurately; only an estimate,

$$z_k = h_k(x_k, v_k) \ , \tag{4}$$

can be made, where h_k is a sensor that maps x_k and a random disturbance v_k, into an observation, z_k. Such problems are solved by reformulating them into a perfect state information problem by introducing an augmented state variable, I_k which is a vector of the past observations and controls applied.

$$\begin{aligned} I_{k+1} &= (I_k, z_{k+1}, u_k), \quad k = 0, 1, \ldots, N-2, \\ I_0 &= z_0 \ . \end{aligned} \tag{5}$$

3 Response Formulation with imperfect State information

In this section will formulate the computer worm response problem as a DP problem with imperfect state information. We assume that there could be only one worm and that the worm is a random scanning worm. We also assume that there is a sensor, such as an IDS albeit not very accurate. This DP formulation tells us which control should be applied to minimize the costs incurred until the worm detection process is complete.

3.1 Problem Statement

System Evolution: Consider a machine that provides some service. This machine needs to be operated for N steps or N time units. This machine can be in one of two states, P or $\overline{P}$, corresponding to the machine being in proper(desired state) or improper(infected by a worm) state respectively. During the course of operating the machine, it goes from state P to $\overline{P}$ with a certain probability λ and remains in state P with a probability $\overline{\lambda} = (1-\lambda)$. If the machine enters state $\overline{P}$, it remains there with probability 1. The infectious force λ, is an unknown and depends on how much of the Internet is infected with the worm, if at all a worm is present.

Sensor: The machine also has a *sensor* which inspects the machine for worm infections. However, it cannot determine the exact state of the machine. Rather, it can only determine the state of a machine with a certain probability. There are two possible observations; denoted by G (good, probably not infected) and B(bad, probably worm infected). Alternatively, instead of infections, we can imagine that the *sensor* looks for infection attempts and anomalies. The outcome would then indicate that there is probably a worm on the Internet(B) or not(G) as opposed to whether the host machine is infected or not. For the time being, let us assume that the inspections happen proactively at random intervals and also when alerts are received from peers. We also assume that the *sensor*'s integrity is not affected by the worm.

Controller: The machine also includes a *controller* that can continue(C) or stop(S) operating the machine. The machine cannot change states by itself if it is stopped. Thus the *controller* can stop the machine to prevent a worm infection and start it when it deems it safe to operate the machine. There are certain costs involved with each of these actions under different conditions as described in the next paragraph. The controller takes each action so that the overall cost of operating the machine for N steps is minimized.

Costs: Continuing(C) to operate the machine when it is in state P costs nothing. It is the nominal. We incur a cost of τ_1 for each time step the machine is stopped(S) irrespective of whether it is infected or not, and a cost τ_2 for each step an infected machine is operated. One might argue that τ_1 and τ_2 should be the same because an infected machine is as bad as a stopped machine. If that argument is true, the problem becomes trivial and it can be stated right away that the most cost effective strategy is to operate the machine uninterrupted until it is infected. On the contrary, we argue that operating an infected machine costs more as it can infect other machines also. Hence, $\tau_2 > \tau_1$.

Alert Sharing Protocol: Since a computer worm is a distributed phenomenon, inspection outcomes at one machine is a valid forecast of the outcome from a later inspection at another identical machine. (This is an assumption we make to develop the formulation and will be relaxed later on when we discuss a practical

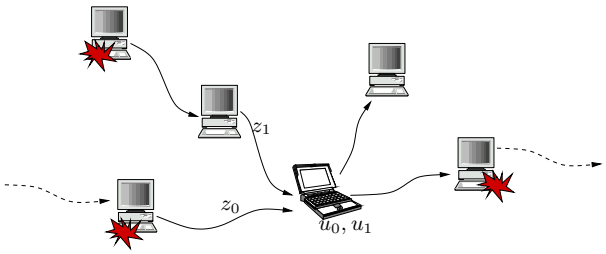

Fig. 1. Alert Sharing Protocol. The laptop is our machine of interest. It uses information, z_0 and z_1, from different chains to choose, actions, u_0 and u_1. It may or may not have seen an anomaly while the machines shown with a blast have seen an anomaly.

application.) Hence, a collection of such machines with identical properties seek to co-operate and share the inspection outcomes. Under this scheme, an inspection outcome at one machine is transmitted to another co-operating peer chosen randomly. The *controller* on the randomly chosen machine uses such received messages to select the optimal control to apply locally. This has the effect of a machine randomly polling several neighbors to know the state of the environment. This gives the uninfected machines an opportunity to take actions that prevent infection. Refer to Fig. 1.

Goal: Now, the problem is to determine the policy that minimizes the total expected cost of operating the machine for N time periods in an environment which could possibly be infected with a worm. DP problems are generally plagued with state space explosion with increasing number of stages to the horizon. However, since we solve the formulation offline, the value of N is not a big constraint unless it is too big. Moreover, once we have the formulation we can also solve it approximately or analytically for larger Ns. The rest of this section develops the formulation for the current problem and provides a solution for $N = 3$. Computer generated results for larger Ns are presented and discussed in later sections.

3.2 Problem Formulation

The above description of the problem fits the general framework of Sect. 2.1, "Problems with imperfect state information." The state, control and observation variables take values as follows:

$$x_k \in \{P, \overline{P}\}, \quad u_k \in \{C, S\}, \quad z_k \in \{G, B\} \ .$$

The machine by itself does not transit from one state to another. Left to itself, it remains put. It is transferred from P to $\overline{P}$ only by a worm infection, a random process – an already infected victim chooses this machine randomly. The evolution of this system follows (1), and is shown in Fig. 2. The function f_k of

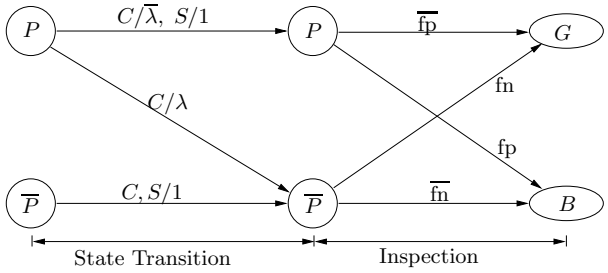

Fig. 2. State Transition probabilities for each action and observation probabilities for each state.

(1) can be derived from Fig. 2 as follows:

$$\begin{aligned} &P(x_{k+1} = P \mid x_k = P,\, u_k = C) = \overline{\lambda}\,, \\ &P(x_{k+1} = \overline{P} \mid x_k = P,\, u_k = C) = \lambda\,, \\ &\vdots \\ &P(x_{k+1} = \overline{P} \mid x_k = \overline{P},\, u_k = S) = 1\ . \end{aligned} \tag{6}$$

The random disturbance, w_k is provided by λ and is rolled in x_k. λ is the infectious force, a function of the number of the machines infected on the Internet. Assuming the machine initially starts in state P, the probability distribution of x_0 is

$$P(x_0 = P) = \overline{\lambda}\,, \qquad P(x_0 = \overline{P}) = \lambda\ . \tag{7}$$

(This assumption is for exposition only. In practice, we do not have to know the initial state the machine starts in.) Recollect that the outcome of each inspection of the machine is an imperfect observation of the state of the system. Thus,

$$\begin{aligned} &P(z_k = G \mid x_k = \overline{P}) = \mathrm{fn}\,, \\ &P(z_k = B \mid x_k = \overline{P}) = (1 - \mathrm{fn})\,, \\ &P(z_k = G \mid x_k = P) = (1 - \mathrm{fp})\,, \\ &P(z_k = B \mid x_k = P) = \mathrm{fp}\ , \end{aligned} \tag{8}$$

where fp and fn are properties of the *sensor*s denoting the false positive and false negative (miss) rates.

Assuming the cost function remains the same regardless of time, the subscript k can be dropped from g_k. We define the cost function as follows:

$$\begin{aligned} &g(P, C) = 0, \qquad g(\overline{P}, C) = \tau_2, \\ &g(P, S) = g(\overline{P}, S) = \tau_1, \\ &g(x_N) = 0. \end{aligned} \tag{9}$$

$g(x_N) = 0$ because u_N is chosen with accurate knowledge of the environment, (i.e) whether there is a worm or not. If there is a worm, $u_N = S$, else $u_N = C$.

Our problem now is to find functions $\mu_k(I_k)$ that minimize the total expected cost

$$\underset{x_k,z_k}{E}\left\{g(x_N)+\sum_{k=0}^{N-1} g\big(x_k,\mu_k(I_k)\big)\right\} \ .$$

We now apply the DP algorithm to the augmented system (refer Sect. 2.1). It involves finding the minimum cost over the two possible actions, C and S, and has the form:

$$\begin{aligned} J_k(I_k) = \min_{\{C,S\}}\Bigg[&\Big(P(x_k = P \,|\, I_k, C)\cdot g(P,C) + P(x_k = \overline{P} \,|\, I_k, C)\cdot g(\overline{P},C)\Big) \\ &\qquad + \underset{z_{k+1}}{E}\Big\{J_{k+1}(I_k, C, z_{k+1}) \,|\, I_k, C\Big\} \ , \\ &\Big(P(x_k = P \,|\, I_k, S)\cdot g(P,S) + P(x_k = \overline{P} \,|\, I_k, S)\cdot g(\overline{P},S)\Big) \\ &\qquad + \underset{z_{k+1}}{E}\Big\{J_{k+1}(I_k, S, z_{k+1}) \,|\, I_k, S\Big\}\Bigg] \end{aligned} \quad (10)$$

where $k = 0, 1, \ldots N-1$ and the terminal condition is $J_N(I_N) = 0$. Applying the costs (9), and noticing that $P(x_k = P \,|\, I_k, S) + P(x_k = \overline{P} \,|\, I_k, S)$ is the sum of probabilities of all elements in a set of exhaustive events, which is 1, we get

$$\begin{aligned} J_k(I_k) = \min_{\{C,S\}}\Bigg[&\tau_2 \cdot P(x_k = \overline{P} \,|\, I_k, C) \ + \underset{z_{k+1}}{E}\Big\{J_{k+1}(I_k, C, z_{k+1}) \,|\, I_k, C\Big\}, \\ &\tau_1 \ + \underset{z_{k+1}}{E}\Big\{J_{k+1}(I_k, S, z_{k+1}) \,|\, I_k, S\Big\}\Bigg] \ . \end{aligned} \quad (11)$$

This is the required DP formulation of response to worms. Next, we demonstrate a solution derivation to this formulation for $N = 3$.

3.3 Solution

Here we show a solution assuming that we expect to know with certainty about the presence of a worm at the receipt of the third message, that is, $N = 3$. The same procedure can be followed for larger Ns.

With that assumption, control u_2 can be determined without ambiguity. If the third message says there is a worm, we set $u_2 = S$, else we set it to C. This also means that the cost to go at that stage is

$$J_2(I_2) = 0 \ . \qquad \text{(Terminal Condition)}$$

Penultimate Stage: In this stage we determine the cost $J_1(I_1)$. Applying the terminal condition to the DP formulation (11), we get

$$J_1(I_1) = \min\Big[\tau_2 \cdot P(x_1 = \overline{P} \,|\, I_1, C) \ , \ \tau_1\Big] \ . \quad (12)$$

The probabilities $P(x_1 = \overline{P} \mid I_1, C)$ can be computed using Bayes' rule and eqs.(6–8), assuming the machine starts in state P. The cost for each of the eight possible values of $I_1 = (z_0, z_1, u_0)$ under each possible control, $u_1 \in \{C, S\}$ is computed using (11). Then, the control with the smallest cost is chosen as the optimal one to apply for each z_1 observed. The *cost-to-go*, $J_1(I_1)$, thus calculated are used for the zeroth stage.

Stage 0*:* In this stage we determine the cost $J_0(I_0)$. We use (11) and values of $J_1(I_1)$ calculated during the previous stage to compute this cost. As before this cost is computed for each of the two possible values of $I_0 = (z_0) = \{G, B\}$, under each possible control, $u_1 = \{C, S\}$. Then, the control with the smallest cost is chosen as the optimal one to apply for the observed state of the machine. Thus we have,

$$J_0(I_0) = \min \Bigg[\tau_2 \cdot P(x_0 = \overline{P} \mid I_0, C) \; + \underset{z_1}{E}\Big\{ J_1(I_1) \mid I_0, C \Big\} \;, \\ \tau_1 + \underset{z_1}{E}\Big\{ J_1(I_1) \mid I_0, S \Big\} \Bigg] \;. \quad (13)$$

The optimal cost for the entire operation is finally given by

$$J^* = P(G)J_0(G) + P(B)J_0(B) \;.$$

We implemented a program that can solve the above formulation for various values of $\lambda, \mathrm{fp}, \mathrm{and\ fn}$. A sample rule-set generated by that program is given in table 1. Armed with this solution, we now show a practical application.

4 A Practical Application

4.1 Optimal Policy

Table 1 shows the optimal policies for a given set of operational parameters. The table is read bottom up. At start, assuming the machine is in state P, the optimal action is to continue, C. At next time step, stage 0, if the observation is B, the optimal action is to stop, S. If $z_0 = B$ is followed by $z_1 = G$, the optimal action is to operate the machine, C. This is denoted by the second line in stage 1. This shows that an undesirable response is backed-off when the environment is deemed not dangerous. In a practical application, such a table will be looked up for a given λ and observation to choose the optimal action. Note that the first, third, sixth and eighth states are unreachable because, for the given z_0, the control u_0 mentioned in the vector is never applied if the system operates in good faith.

4.2 Choosing λ

The value of λ varies with the extent of infection in the Internet. Given that we are uncertain that there is a worm in the Internet, λ cannot be determined

Table 1. An optimal policy table

$\lambda = 0.50, \quad \text{fp} = 0.20, \quad \text{fn} = 0.10$ $\tau_1 = 1, \quad \tau_2 = 2$			
	I_k	J_k	u_k
Stage 1	(G, G, S)	0.031	C
	(B, G, S)	0.720	C
	(G, B, S)	0.720	C
	(B, B, S)	1.000	S
	(G, G, C)	0.270	C
	(B, G, C)	1.000	S
	(G, B, C)	1.000	S
	(B, B, C)	1.000	S
Stage 0	(G)	0.922	C
	(B)	1.936	S
Start		1.480	C

with any accuracy. Rather, only estimates can be made. Hence the distributed Sequential Hypothesis Testing developed earlier is used to estimate λ [6].

Given a sequence of observations $\boldsymbol{y} = \{y_0, y_1, \ldots, y_n\}$, made by a sequence of other participating nodes, and two contradicting hypotheses that there is a worm on the Internet(H_1) and not(H_0), the former is chosen when the likelihood ratio $L(\boldsymbol{y})$ of these hypotheses is greater than a certain threshold η [6]. This threshold η is determined by the performance conditions required of the algorithm. Assuming the observations are independent, $L(\boldsymbol{y})$ and η are defined as follows:

$$L(\boldsymbol{y}) = \prod_{i=1}^{n} \frac{P(y_i|H_1)}{P(y_i|H_0)}, \quad \eta = \frac{DD}{DF} \ , \tag{14}$$

where DD is the minimum desired detection rate and DF is the maximum tolerable false positive rate of the distributed Sequential Hypothesis Testing(dSHT) algorithm. We define each of the above probabilities as follows:

$$\begin{aligned} P(y_k = B \mid H_1) &= [\lambda\,(1 - \text{fn}) + (1 - \lambda)\,\text{fp}]\,, \\ P(y_k = G \mid H_1) &= [(\lambda\,\text{fn}) + (1 - \lambda)(1 - \text{fp})]\,, \\ P(y_k = B \mid H_0) &= \text{fp}\,, \\ P(y_k = G \mid H_0) &= (1 - \text{fp}) \ . \end{aligned} \tag{15}$$

The first one in the above set is the probability of observing a B given hypothesis H_1 is true is the sum of probability of getting infected (λ) times the probability of detection, and the probability of not getting infected($1-\lambda$) times the probability of false positives. The others in (15) are defined similarly.

For any given sequence of observations, we calculate $L(\boldsymbol{y})$ for several values of λ – say for ten different values in steps of 0.1 starting at 0.1. The lowest λ for which the $L(\boldsymbol{y})$ exceeds η will be taken as the current levels of infection and used in determining the optimal response. The reason for choosing discrete values of λ will be apparent shortly.

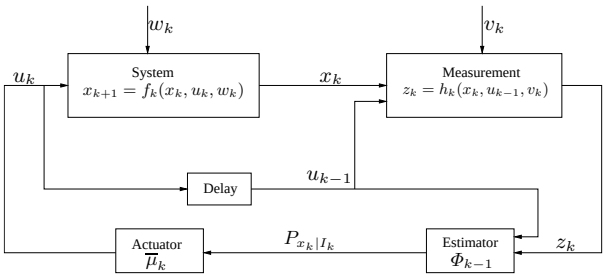

Fig. 3. The controller split into an *Estimator* and an *Actuator*

Given (14) and (15), all observations over a sequence of nodes can be expressed as one number, the $L(\boldsymbol{y})$. A node receiving this number from a neighbor, can update it using its own observations and (14), to estimate λ.

In practice however, a table of rule-sets is calculated offline for several candidate values of λ. Then, the table corresponding to the λ as chosen above is consulted to choose u_k given I_k . Thus, each node only receives a likelihood ratio of the worm's presence from its peers. Each node also has to only remember its own past observations and corresponding actions(I_k).

4.3 Larger Ns

As N increases, the dimensions of I_k increases which in turn increases the number of the calcuations involved exponentially. This problem can be overcome by reformulating the problem to represent the state of the system probabistically based only on the last observation and control applied. In other words, we reduce I_k to smaller dimensions containing only the *Sufficient Statistics* yet summarizing all essential contents of I_k as far as control is concerned. Figure 3 explains this concept. The estimator Φ_{k-1} estimates the probalistic state of the system $P_{x_k|I_k}$ based on $P_{x_{k-1}|I_{k-1}}$, z_k and u_{k-1} . The actuator, $\overline{\mu}_k$, then selects the optimal response based on $P_{x_k|I_k}$.

This re-formulation makes it easy to apply the response model for larger Ns. We implement this model and evaluate it in a simulation. The evaluation and the results are discussed in the next section.

5 Evaluation

The sufficient statistics formulation discussed in the previous section was implemented and evaluated with a discrete event simulation. The simulation consisted of 1000 participants with 10% of the machines being vulnerable. We set the number of stages to operate the machine, $N = 4$ to calculate the rule-sets. Note that $N = 4$ is used only to calculate the rule-sets but the machines can be operated for any number of steps. N is essentially the number of past observations and actions that each machine remembers. The local IDSes were set to have a

false positive and false negative rates of 0.1. These characteristics of the local IDS is used to calculate the probability of infection, λ with a desired worm detection rate of 0.9 and failure rate of 0.1. In all the following experiments, we used a random scanning worm which scans for vulnerable machines once every unit-time.

5.1 Experiments

Parameters of Evaluation: A set of experiments was designed to understand the effect of various parameters on the effectiveness of the model in controlling the spread of the worm. The only free variable we have here is the ratio τ_2/τ_1. There is no one particular parameter that can measure or describe the effectiveness of the response model. Rather, the effectiveness is described by a pair of parameters – numbers of machines that are not infected and that provide service, i. e. in state C.

Algorithm: The algorithm for the discrete-event simulation is as follows. At each time cycle,

- all infected machines attempt one infection,
- all machines that had a alert to share, share the likelihood ratio that there is a worm on the Internet with one another randomly chosen vulnerable node,
- and all vulnerable machines that received an alert earlier take a response action based on the information received and the current local observations.

Results: In the first experiment, we want to make sure that we have a worm that behaves as normal random scanning worm and validate the response model for the degenerate cases. We verify this by providing no response. This response can be achieved by setting the cost ratio to 1 – the cost of stopping the service is the same as getting infected. In this scenario, we expect the response model not to take any defensive measures against suspected infection attempts. As expected, we see in Fig. 4, that none of the machines are stopped (S state). The worm spreads as it would spread when there is no response in place. This validates our worm and also our response model.

As another sanity check we set the machines to remember infection attempts forever. Under this policy, once a machine enters the S state, it remains in that state forever. We see that in this case (Fig. 5) the number of machines infected are very low except when $\tau_2/\tau_1 = 1$.

In the next experiment, we try to understand the behavior of our response model in various situations. Since the only free variable is the ratio τ_2/τ_1, we repeat the previous experiment with various values for that ratio. The results for this set of experiments is shown in Fig. 6. This graph shows behavior of our response model in five different tests. There are two different curves for each test indicating the number of vulnerable machines being infected and the number in S state against time. We can see that when the ratio is 1, the number machines that are in S state is 0. As the ratio τ_2/τ_1 rises, the response becomes stricter.

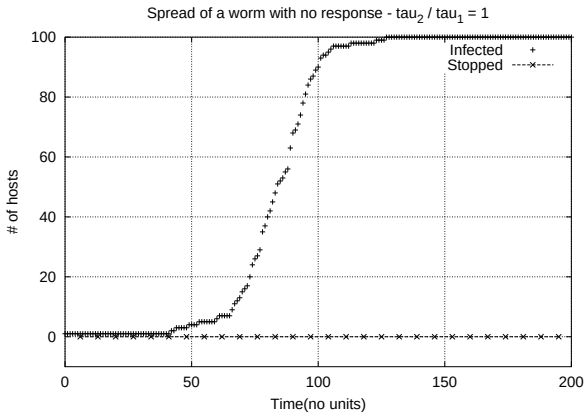

Fig. 4. No machines are stopped when the cost of being infected is the same as cost of shutting down the machine.

We see that the number of machines in the stopped(S) state is higher when the cost of being infected is higher. Also the worms spreads significantly slower than without any response in place or with a lower τ_2/τ_1 ratio.

5.2 Effects of increasing N

The experiments shown earlier in this section were all conducted with $N = 4$. An interesting question to ask here, "What happens if we increase the value of N?". Fig. 7 shows the performance of the system for various values of N while holding the ratio of τ_2/τ_1 constant at 30. The set of sigmoidal curves trace the growth of the worm, while the other set of curves trace the number of nodes that are shut-down at any given instance of time. We notice that there is no appreciable slowing of the worm with increased values of N – all the worm growth curves are bunched up together. This is due to the small number of dimensions to the state, $x_k \in \{P, \overline{P}\}$. A larger observation space does not contribute much to improve the performance of the system.

6 Strengths, Limitations and Future Work

There are several topics in this paper yet to be addressed. There are issues to be addressed from three different perspectives – one, problems that would arise during the practical adoption of this model, two, in the evaluation, and three, in the model itself.

This is a collaborative response model. As with any collaborative effort, there is a host of issues such as privacy, confidentiality, non-repudiation, etc, that will need to be addressed during practical adoption. Thankfully, these are issues for which there are solutions available already through cryptography and IPSEC. In

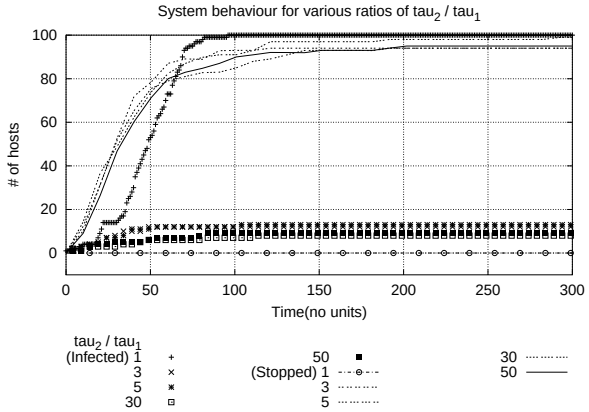

Fig. 5. Once entered the S state, a machine stays there.

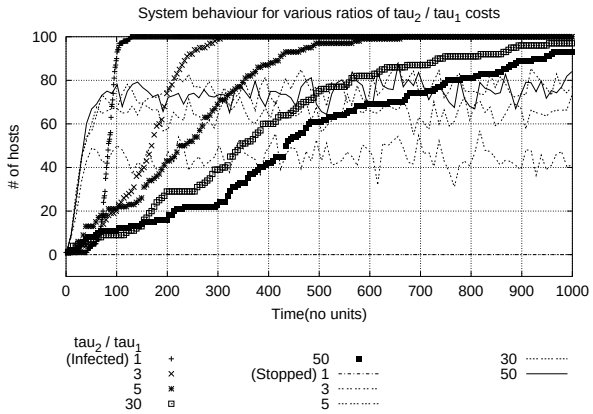

Fig. 6. Higher costs of being infection invoke stricter responses.

a co-operative or collaborative environment, we expect these issues to be either already addressed. Still, co-operation amongst various entities on the Internet such as amongst different corporate networks pose more legal and economic problems than technical. In such cases where sharing anomaly information with networks outside of the corporation is not feasible, applying this response model within the corporate network itself can provide valuable protection.

Another major adoption problem is the assignment of realistic values to the costs τ_1 and τ_2. However, there is prior work that attempts to assign costs to various responses that can used [3, 11].

Evaluating worm defenses is a difficult problem [7]. At one extreme we have realistic evaluation possible only on the Internet but is infeasible. At the other extreme, we have mathematical models only. In between these two extremes, we

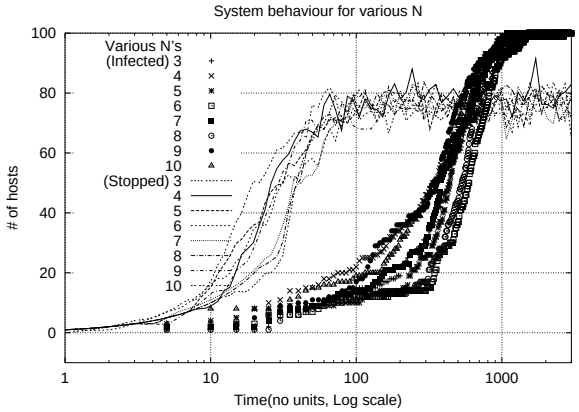

Fig. 7. Larger Ns do not contribute much to improve performance.

have simulations such as the one used in this paper and emulation as used in a study for worm detection [6].

With the availability of data about Internet traffic during worm outbreaks, it may be possible to evaluate the defense model on a network testbed such as Emulab [19] by replaying the traffic for a scaled down version of the Internet. Such an experiment would need the available data to be carefully replayed with tools such as TCP Replay,TCP Opera, etc. This is a task that can explored in the future to evaluate worm defenses. Scaling down the Internet is another problem in itself [18].

An issue to be studied is the behaviour of this model in face of false alarms and isolated intrusions. For eg., consider one and only participant raising an alarm for an isolated event and several other participants choosing the S control. We would like to know when these participants would apply the C control. Trivially, we can set a time-out for the defense to be turned-off. However, the time-out should be chosen carefully and probably be dynamic to guard against exposing oneself to slow worm attacks.

When there is a cost to sampling packets, this model can be extended to optimally stop the sampling process and declare either that there is a worm or that there is no worm. Interestingly, this extension would lead us to the distributed Sequential Hypothesis Testing that we discussed in our previous paper [6]. Actions such as C and S if applied frequently could lead to a very unstable system. We need to evaluate this factor in light of ambient anomaly levels in different environments. This is a problem with the model itself. However, this can be alleviated during adoption in various ways. For example, the set of response options can be made larger with options to have several levels of reduced functionality instead completely shutting down the service.

When all participants behave identically each participant knows exactly how the others will behave. In such a scenario, each one can make a better decision

about the optimal control to be applied taking into account the others' behavior. For example, if participant A determines that the optimal policy to be applied is S, it now knows that all other participants will also apply the same control. Then, there is no need for A to apply S. Instead A could apply C as there is no opportunity for a worm to spread when all others participants are stopped. The problem now enters the realm of *game theory*.

6.1 Strength of this technique

One question that needs to be answered for any defensive technique is this: "If the attacker knows about the approach being used for defense, will (s)he be able to write a new generation of worms that can overcome the defense?"

There are two different goals that an attacker with knowledge about our system can try to achieve. One, try to circumvent the defense and spread the worm. Two, trick the defense into over-reacting.

The second goal cannot be achieved because of the dynamic and self-regulating nature of our approach which is based on the current environmental conditions as depicted in Fig. 3. The attacker may force our system to react to an initial stimulus which is not a true worm but once the stimulus has reduced, the defence pulls back too.

To achieve the first goal, the worm needs to either spread very slowly such that information about anomalous incidents are forgotten by the participants, or attack pre-selected victims that may not be alerted by its peers. However, since the alerts are shared with randomly chosen peers while the worm is spreading, there can be no effective pre-selection that can overcome the defense. Whereas a slow spreading worm might be successful to a certain extent.

Nevertheless, we believe that a slow spreading worm can be identified by other means such as manual trouble-shooting prompted by the ill-effects of the worm; unless the worm installs a time-bomb that is set to trigger after the worm has spread to most vulnerable nodes. We also believe that such slow worms will be circumvented by routine maintenance patches – most worms we know so far have exploited only known, but unpatched, vunlerabilities.

Moreover, there is a heightened awarness of security issues amongst the information technology community than ever before. Laws related to data security are being tightened and enforced more vigorosly than in the past. Patch generation and deployment techniques have advanced tremendously recently. In such an environment, we expect that steps to patch or workaround known vulnerabilities will be taken with more urgency than ever before effectively thwarting extremely slow worms discussed in the preceeding paragraphs.

Thus, the worm has a very narrow window between spreading too slow and spreading too fast – the window where our response mechanism works to thwart the worm. In conclusion, knowledge of our approach does not provide much value to the attacker or new generation of worms.

References

1. Kostas G. Anagnostakis et al. A cooperative immunization system for an untrusting internet. In *Proc. of the 11th IEEE ICON*, pages 403–408, October 2003.
2. Kostas G. Anagnostakis, Michael B. Greenwald, Sotiris Ioannidis, and Angelos D. Keromytis. Robust reactions to potential day-zero worms through cooperation and validation. In *Proc. of the 9th Information Security Conference (ISC).*, 2006.
3. Ivan Balepin, Sergei Maltsev, Jeff Rowe, and Karl Levitt. Using specification-based intrusion detection for automated response. In *Proc. of RAID*, Pittsburg, 2003.
4. Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific, third edition, 2005.
5. Min Cai, Kai Hwang, Yu-Kwong Kwok, Shanshan Song, and Yu Chen. Collaborative internet worm containment. *IEEE Security and Privacy*, 4(3):34–43, May 2005.
6. S. G. Cheetancheri et al. A distributed host-based worm detection system. In *Proc. of SIGCOMM LSAD*, pages 107–113. ACM Press, 2006.
7. S. G. Cheetancheri et al. Towards a framework for worm defense evaluation. In *Proc. of the IPCCC Malware Workshop on Swarm Intelligence*, Phoenix, Az, April 2006.
8. Manuel Costa et al. Vigilante: end-to-end containment of internet worms. In *Proc. of the SOSP*, pages 133–147. ACM Press, 2005.
9. Denver Dash et al. When gossip is good: Distributed probabilistic inference for detection of slow network intrusions. In *Proc. of the AAAI*, 2006.
10. Hyang-Ah Kim and Brad Karp. Autograph: Toward automated, distributed worm signature detection. In *Proc. of the USENIX Security Symposium*, 2004.
11. Wenke Lee, Wei Fan, Matthew Miller, Salvatore J. Stolfo, and Erez Zadok. Towards cost-sensitive modeling for intrusion detection and response. In *J. of Computer Security*, volume 10, 2002. Numbers 1,2.
12. David J. Malan and Michael D. Smith. Host-based detection of worms through peer-to-peer cooperation. In *Proc. of the WORM*, pages 72–80. ACM Press, 2005.
13. James Newsome, Brad Karp, and Dawn Song. Polygraph: Automatically generating signatures for polymorphic worms. In *Proc. of the IEEE Symposium on Security and Privacy*, pages 226–241. IEEE, 2005.
14. Sumeet Singh, Cristian Estan, George Varghese, and Stefan Savage. Automated worm fingerprinting. In *Proc. of the Sixth OSDI*, San Francisco, CA, December 2004.
15. S.Sidiroglou and A D Keromytis. Countering network worms through automatic patch generation. *IEEE Security and Privacy*, 3(6):41 – 49, November 2005.
16. Ke Wang, Gabriela Cretu, and Salvatore J. Stolfo. Anomalous payload-based worm detection and signature generation. In *Proc. of the RAID*. ACM Press, 2005.
17. Ke Wang and Salvatore J. Stolfo. Anomalous payload-based network intrusion detection. In *Proc. of the RAID*. ACM Press, Sept 2004.
18. Nicholas Weaver, Ihab Hamadeh, George Kesidis, and Vern Paxson. Preliminary results using scale-down to explore worm dynamics. In *Proc. of the WORM*, pages 65–72. ACM Press, 2004.
19. Brian White et al. An integrated experimental environment for distributed systems and networks. In *OSDI*, pages 255–270, Boston, MA, December 2002. USENIX.
20. Cliff Changchun Zou, Lixin Gao, Weibo Gong, and Don Towsley. Monitoring and early warning for internet worms. In *Proc. of the CCS*, pages 190–199. ACM Press, 2003.