Analysis of a Markov Decision Process Model For Intrusion Tolerance

O. Patrick Kreidl
MIT Laboratory for Information and Decision Systems
Cambridge, MA 02139 U.S.A.
opk@mit.edu

Abstract

We consider a simplest Markov decision process model for intrusion tolerance, assuming that (i) each attack proceeds through one or more steps before the system's security fails and (ii) defensive responses targeting these intermediate steps may only sometimes thwart the attack. Our analysis shows that, even in the ideal case of perfect detectors, it can be sub-optimal in the long run to employ defensive responses while under attack; that is, depending on attack dynamics and response effectiveness, the total overhead of ongoing defensive countermeasures can exceed the total risk of intermittent security failures. Simulation experiments reveal that a tradeoff between these two types of costs persists in the realistic case of imperfect detectors. These experiments also shed light on (i) the extent to which increasing sensor uncertainty monotonically degrades achievable performance and (ii) the loss from optimum performance of two popular rule-based policies for response selection.

1. Introduction

Intrusion tolerance is an active area of research that aims to extend proven solutions in the fields of fault-tolerance and dependability to address growing concerns about weaknesses of modern-day information security systems [1]. The key assumption is that, even when state-of-the-art intrusion prevention and detection techniques are in place, an attacker will eventually break through. An intrusion tolerant system (see Fig. 1) is therefore also equipped with mechanisms for reacting in real-time to an ongoing intrusion, striving not just to detect its occurrence but also to diagnose or predict its evolution so that defensive countermeasures can be invoked timely enough to delay, frustrate or altogether thwart the attacker's ultimate intentions. Successful realization of this vision introduces numerous technical challenges, including the embedded design of real-time sensing/response devices as well as the automatic control of these devices to favorably influence the system's security status over time.

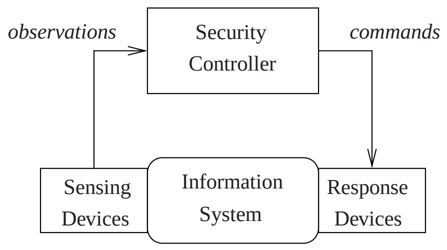
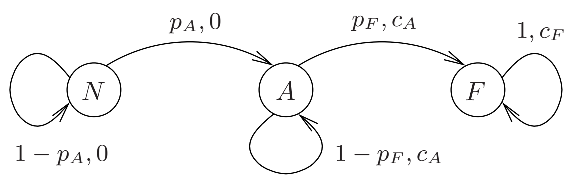


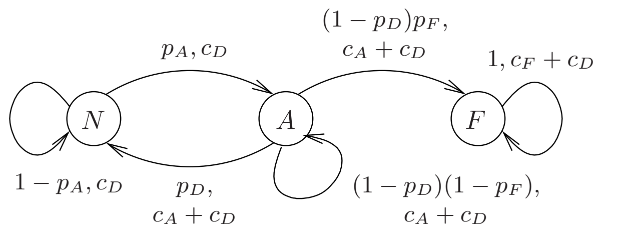
Figure 1. An Intrusion Tolerant System

This paper analyzes the problem of intrusion tolerance within the formalism of Markov Decision Processes (MDPs) [2]. MDPs are related to the types of stochastic models already popular in contemporary dependability/security research [3, 4, 5, 6]. Their focus is to assume that the model parameters (e.g., state transition probabilities & costs) in each decision stage depend explicitly on the control action taken in that stage. For example, an MDP can capture the reinstatement of security (and the possible cost of disrupting all normal services) upon taking a cleansing ”reset” action (e.g., initiating a server rotation [7] or perhaps a more elaborate recovery sequence [8]); as another example, an MDP can capture the possible preemption of a security failure from a well-timed ”defend” action (e.g., killing a process that is critical to the intrusion's continuation [9]). In any case, the expected performance (e.g., mean-time-to-failure) is dependent on the *policy*, or rule that dictates how control actions are selected, and the *optimal policy* is one that achieves the best performance for a given model.

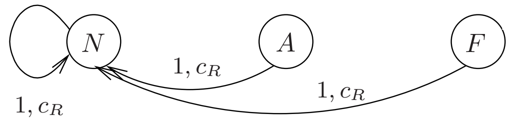
Section 2 presents in detail a simplest MDP model for intrusion tolerance problems. Though the model assumes only three system states and only three candidate controls for ease of analysis, it is sufficiently rich to expose a number of important policy considerations for intrusion tolerant systems. In particular, even in the ideal case of perfect detectors (Subsection 3.1), it is sometimes sub-optimal in the long run to employ a policy that invokes defensive responses during the intermediate stages of an intrusion; more specifically, depending on attack dynamics and response effectiveness, policies involving only occasional reset actions may provide a better tradeoff between the total overhead of ongoing security maintenance and the total risk of potential security failures. The tradeoff between these two types of penalties is shown through simulation experiments (in Subsection 3.2) to persist in the realistic case of imperfect detectors. Moreover, these experiments evaluate empirically (i) the extent to which increasing sensor uncertainty monotonically degrades achievable performance and (ii) the loss from optimum performance of two popular rule-based policies for selecting responses. Despite the model simplicity, these rule-based policies are seen to incur up to 35% additional costs to those incurred by the optimized policy—an even greater percentage is conjectured for intrusion tolerance models of more realistic size and character.



(a) Under a "Wait" action (Control W)



(b) Under a "Defend" Action (Control D)



(c) Under a "Reset" Action (Control R)

Figure 2. A three-state Markov model (a) under a "wait" action, in which case a system operating normally (i.e., in state N) will eventually fall under attack and subsequently experience a security failure (i.e., transition to state A and then to state F); (b) under a "defend" action, which possibly prevents an ongoing attack from causing a security failure; and (c) under a "reset" action, which always reinstates the system to normal operation.

2. MDP Model for Intrusion Tolerance

This section abstracts the problem of intrusion tolerance within the modeling formalism of a stationary, discrete-time, finite-state, finite-action Markov Decision Process (MDP) [2]. Underlying such a model is (i) a finite collection of finite-state Markov chains, one per candidate control action, and (ii) the assumption that decisions are made in discrete stages: each stage begins in a particular state and, upon taking a control action, the process probabilistically transitions to a new state (or self-transitions to the same state), simultaneously ending the current stage and beginning the next. Thus, the outcome of each decision stage is defined by the selected control and the realized state transition. In turn, an MDP model associates both a probability and a cost to every such outcome.

We categorize the security environment into only three states, labeled N, A and F to denote "operating normally," "under attack" and "security failure," respectively; there are similarly only three candidate controls, labeled W, D and R to denote a "wait" action, a "defend" action, and a "reset" action, respectively. The three diagrams in Fig. 2 summarize the single-stage transition probabilities/costs under each control action. For example, under the "wait" action (control W), self-transitions in state N represent secure, normal operation of the information system; however, a transition from state N to A occurs with per-stage probability p_A, representing the beginning of an intrusion attempt. Thereafter, self-transitions in state A represent a secure system being under attack; eventually, a transition from state A to F occurs with per-stage probability p_F, representing the beginning of a security failure that persists indefinitely. Stages that begin in state N are cost-free, whereas stages that begin in state A incur a cost of c_A and self-transitions in state F incur a cost of c_F. These same probability/cost parameters apply under the "defend" action (control D), but with two differences: firstly, the possible transition from state A back to N (occurring with per-stage probability p_D) represents the successful disruption of the intrusion attempt; secondly, a cost of c_D is incurred in addition to the transition cost under control W. Finally, under the "reset" action (control R), a transition back to state N occurs with probability one, incurring a cost of c_R, no matter the state at the beginning of the decision stage.

Summarizing, our simplest MDP model is described by three probability parameters and four cost parameters. The probability parameters may take values strictly between zero and unity; in particular, no value of one probability restricts the value of another. This reflects the fact that each probability parameter abstracts a different characteristic of the intrusion tolerance problem. More specifically, probability p_A relates to the frequency with which intrusion attempts are initiated, where larger values pertain to more aggressive attackers. Probability p_F relates to the rate at which an initiated intrusion will penetrate the security system, where larger values pertain to more vulnerable systems (or perhaps just shorter-duration exploits). Probability p_D relates to the success rate of a well-timed defensive countermeasure in preventing an otherwise inevitable security failure, where larger values pertain to more effective response devices. The cost parameters may take any set of non-negative values that satisfy the following inequalities:

- $c_A < c_F$, meaning that one stage of a security failure incurs more cost than one stage of an intrusion attempt;
- $c_D < c_R$, meaning that the reset action incurs more maintenance cost (because of e.g., a greater disruption to normal services) than the defend action; and
- $c_R \leq c_F$, meaning that one stage of a security failure incurs no less cost than invoking the reset action.

3 Analysis of the Optimal Policy

Subject to design in a MDP is the policy by which, in each stage, all available information maps into each selected control, typically seeking to minimize the average-cost-per-stage assuming a large number of stages. We now proceed to analyze the optimal policy for the simple MDP model described in Section 2. Subsection 3.1 will begin under the idealization of perfect detectors, in which case the optimal policy admits an analytical solution: its equivalence to the most natural policy (i.e., taking actions W, D and R when in states N, A and F, respectively) holds only for a subset of the possible parameter values. Subsection 3.2 continues the analysis for the case of imperfect detectors, evaluating alternative policies via a combination of numerical algorithms and Monte-Carlo simulations. Altogether, the analyses capture the inherent tradeoff between risk due to security failures and overhead due to defensive countermeasures; moreover, a control policy that fails to account for all model parameters is also likely to fail to maintain the best ”long-run” balance between these two types of penalties.

3.1 The Case of Perfect Detectors

Let us assume perfect detectors for now: mathematically, in each stage $k = 0, 1, \ldots$, the current state $x_k \in \{N, A, F\}$ is revealed before the control decision $u_k \in \{W, D, R\}$ must be made. Of course, the next state x_{k+1} remains uncertain until action u_k is applied and the process proceeds to the subsequent decision stage. Let $g(x_k, u_k, x_{k+1})$ denote the cost associated with the outcome of stage k e.g., in our model, it may be any one of the different transition costs provided in Fig. 2. We desire a control policy that minimizes the *infinite-horizon average-cost-per-stage*

$$\lambda = \lim_{T\to\infty} \left(\frac{1}{T} \sum_{k=0}^{T} g(x_k, u_k, x_{k+1}) \right),$$

which under certain technical conditions (satisfied by our model) is known to be achievable with a *stationary*, or stage-invariant, policy [2]. This means, in our model, that the optimal policy is one of the twenty-seven functions of the form $\mu : \{N, A, F\} \to \{W, D, R\}$, each a different mapping from the state space to the control space by which every single-stage decision $u_k = \mu(x_k)$ is to be made.

Given a particular policy μ, equations are provided in [2] to derive the associated average-cost-per-stage λ^μ in terms of the given model parameters. However, under the assumptions on cost parameters stated in the previous subsection, it is easily seen that most of the twenty-seven candidate policies may be quickly dismissed. Firstly, because the only hope of exiting the failure state is by invoking the reset action (and $c_R \leq c_F$), we need only consider policies that map state F to action R. Secondly, because the defend action has no bearing on transitions out of normal operation, we need *not* consider policies that map state N to action D. This leaves only six candidate policies, three mapping state N to the wait action W and three mapping state N to the reset action R—let us first evaluate the former three policies, each taking a different action when in state A.

1. *Wait-upon-attack* μ^W, or taking action W when in state A and incurring cost c_A per stage until the transition to state F occurs, achieving
$$\lambda^W = \frac{p_A(c_A + p_F c_R)}{p_F(1 + p_A) + p_A}. \tag{1}$$
2. *Defend-upon-attack* μ^D, or taking action D when in state A and incurring cost $c_A + c_D$ per stage until the transition to state N or to state F occurs, achieving
$$\lambda^D = \frac{p_A\left[c_A + c_D + (1 - p_D)p_F c_R\right]}{p_F(1 + p_A) + p_A + p_D\left[1 - p_F(1 + p_A)\right]}. \tag{2}$$
3. *Reset-upon-attack* μ^R, or taking action R immediately upon exiting state N, achieving
$$\lambda^R = \frac{p_A c_R}{1 + p_A}. \tag{3}$$

The remaining policies, namely those mapping state N to action R, each achieve an average-cost-per-stage of c_R and can thus be dismissed by virtue of (3).

First notice that comparing the policies (1)-(3) in terms of the Mean-Time-To-Failure (MTTF) would not fully capture the “long-run” cost tradeoffs. A failure occurs on-average once every $1 + 1/p_A + 1/p_F$ stages under policy μ^W and once every $1 + 1/p_A + (p_A + p_D)/[p_A p_F(1 - p_D)]$ stages under policy μ^D. Moreover, under policy μ^R, the MTTF is technically infinite but, proactively, the reset action is invoked on-average once every $1 + 1/p_A$ stages. So, regardless of the values of the three probability parameters, policy μ^D results in the longest MTTF. In contrast, it is readily seen that the ordering of (1)-(3) also depends on the values for cost parameters c_A, c_D and c_R. The comparisons are even more subtle: consider, for example, the comparison between policies μ^W and μ^D in terms of their average costs. While the numerator of λ^W in (1) is always greater than or equal to the numerator of λ^D in (2), the same cannot be said for the respective denominators. In particular, the denominator of λ^W is less than that of λ^D if and only if the probability p_F is less than $1/(1+p_A)$. In general, the ordering of (1)-(3) depends on all model parameters, the probabilities as well as the costs.

It is also worth noting a few limiting properties of (1)-(3). Firstly, if probability p_D approaches zero, meaning the defend action is nearly ineffective against an ongoing intrusion, then $\lambda^D \geq \lambda^W$ with equality only if cost c_D is also zero. Secondly, $\lambda^R < \lambda^W$ if and only if the cost ratio c_A/c_R exceeds the fraction $p_A/(1+p_A)$. This is arguably surprising: despite its appearance in (1), probability p_F has no bearing on the comparison between policy μ^W and policy μ^R. In particular, the ordering of λ^W and λ^R is unchanged even as p_F approaches unity, or the case where each intrusion attempt almost immediately results in a security failure (e.g., a flash worm [10]). In summary, assuming an ineffective defend action (in which case policy μ^D is sub-optimal), policy μ^R is optimal only in the case of sufficiently infrequent attacks (i.e., only for small p_A such that $p_A/(1+p_A) < c_A/c_R$ is satisfied)—in the case of more frequent attacks, policy μ^W is optimal.

The preceding comparison between λ^W and λ^R can be expressed concisely by the inequalities

$$\lambda^R \quad \underset{\text{choose } \mu^R}{\overset{\text{choose } \mu^W}{\gtrless}} \quad \lambda^W \quad \Longleftrightarrow \quad \frac{p_A}{1+p_A} \quad \underset{\text{choose } \mu^R}{\overset{\text{choose } \mu^W}{\gtrless}} \quad \frac{c_A}{c_R}\ . \tag{4}$$

The general characterization of the optimal policy also includes the other two pairwise comparisons of (1)-(3). The comparison between λ^D and λ^R reduces to the inequalities

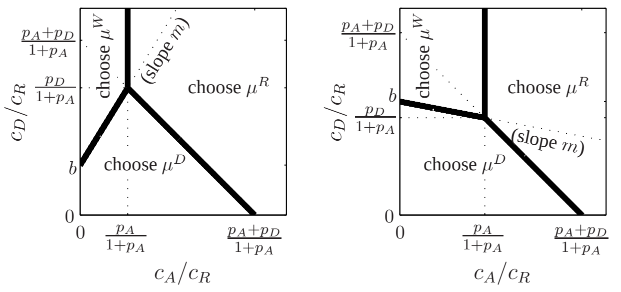
$$\frac{c_D}{c_R} \quad \underset{\text{choose } \mu^D}{\overset{\text{choose } \mu^R}{\gtrless}} \quad -\frac{c_A}{c_R} + \frac{p_A+p_D}{1+p_A}\ , \tag{5}$$

while the comparison between λ^W and λ^D reduces to

$$\frac{c_D}{c_R} \quad \underset{\text{choose } \mu^D}{\overset{\text{choose } \mu^W}{\gtrless}} \quad m\frac{c_A}{c_R} + b \tag{6}$$

with slope m and intercept b, respectively, given by

$$m = \frac{p_D\left[1 - p_F(1+p_A)\right]}{p_F(1+p_A)+p_A} \quad \text{and} \quad b = \frac{p_D p_F(1+p_A)}{p_F(1+p_A)+p_A}\ .$$



(a) Infrequent/Slow Attacks (b) Frequent/Fast Attacks

Figure 3. The partitions characterizing the optimal policy for the MDP model in Fig. 2.

Observe that probability p_A affects all three pairwise comparisons, while probability p_D affects only the latter two comparisons and probability p_F affects only the last comparison. Viewing the (normalized) costs c_A/c_R and c_D/c_R as degrees-of-freedom, (4)-(6) form a system of linear inequalities with coefficients depending on the values of probabilities p_A, p_F and p_D. In particular, the set of all normalized costs comprises the unit square, and each inequality alone defines a partition of this unit-square in accordance with which of the two associated policies is preferable. In turn, all three inequalities together define the partition in accordance with which of the three policies is optimal.

Fig. 3 graphically summarizes these optimal partitions, showing them for different assumptions on attack dynamics. Consistent with intuition, the optimal policy is (i) to wait-under-attack when the attack cost c_A is small but the defend cost c_D is large, (ii) to defend-under-attack when both c_D and c_A are small and (iii) to reset-under-attack when both c_A and c_D are large. Less intuitive is what the optimal policy is when either of these costs takes a mid-range value, and particularly how this depends on the probability parameters. For instance, notice first that (i) policy μ^D cannot be optimal if c_A/c_R exceeds $(p_A+p_D)/(1+p_A)$ and (ii) the three regions always meet at the point where $c_A/c_R = p_A/(1+p_A)$ and $c_D/c_R = p_D/(1+p_A)$. Indeed, the region for policy μ^D reduces in size as p_D decreases (i.e., as the defend action becomes less effective), vanishing entirely as p_D approaches zero. Next, as probability p_F increases (i.e., as attacks more quickly bring about failure), the boundary between policies μ^W and μ^D pivots about the central meeting point as slope m decreases. Moreover, the sensitivity of the optimal partition to such changes in probabilities p_D or p_F is also a function of probability p_A (i.e., of the frequency with which attacks occur).

For a particular choice of probability parameters, Fig. 4(a) plots the achieved average-cost-per-stage λ^* (normalized by the reset cost c_R). It is seen to increase lin-

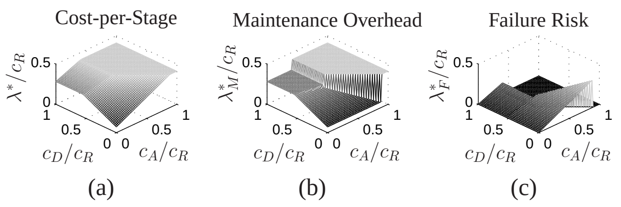


Figure 4. The (a) performance of the optimal policy in Fig. 3(b), taking $p_A = 0.7$, $p_F = 0.9$ and $p_D = 0.8$, and its tradeoff between (b) maintenance overhead and (c) failure risk.

early as either c_A or c_D increases *unless* these costs are such that policy μ^R is optimal, in which case λ^* plateaus at λ^R. Fig. 4(b) and Fig. 4(c) illustrate that λ^* is, in general, the sum of two types of security-related penalties. Here, we have denoted the maintenance overhead by λ^*_M, which is the sum of terms involving costs c_D or c_R, and the failure risk by λ^*_F, which is the sum of terms involving costs c_A or c_F.[1] For example, λ^W in (1) decomposes into the sum of

$$\lambda^W_M = \frac{p_A p_F c_R}{p_F(1+p_A)+p_A} \text{ and } \lambda^W_F = \frac{p_A c_A}{p_F(1+p_A)+p_A},$$

whereas λ^R in (3) decomposes into $\lambda^R_M = \lambda^R$ and $\lambda^R_F = 0$.

3.2 The Case of Imperfect Detectors

The analysis thus far assumes that each control decision has access to perfect state information: the detectors never generate a false-positive nor a false-negative (i.e., never indicate state A if state N is true nor state N if state A is true, respectively) and, similarly, the presence or absence of a security failure is never in question (i.e., state F is indicated if and only if state F is true). To extend the analysis to the realistic case of imperfect state information, we turn to the more general formalism of a *Partially Observable* Markov Decision Process (POMDP) [11]. In addition to parameters of the MDP, a POMDP model assumes that each observation takes one of a finite set of values and then includes probability parameters that relate to sensor uncertainty. Our three-state model, for instance, could include (at least[2]) two additional parameters: a false-positive probability $q_{A|N}$ and a false-negative probability $q_{N|A}$; notice that the ideal case is recovered when these probabilities are both zero.

The advent of imperfect state information has dramatic ramifications on the structure of the optimal policy and, in turn, the complexity of its analysis and computation. This stems from the fact that, in each stage $k = 0, 1, \ldots$, it is no longer true that state x_k is revealed before the control decision u_k is made; rather, only an observation z_k is revealed and it must be properly interpreted within the context established from *all* previously revealed information i.e., the observation history $\mathbb{Z}_{k-1} = (z_0, z_1, \ldots, z_{k-1})$ as well as the control history $\mathbb{U}_{k-1} = (u_0, u_1, \ldots, u_{k-1})$. Under certain technical conditions (satisfied by POMDPs), this interpretation is provided by the conditional probability distribution $P(x_k|\mathbb{Z}_k, \mathbb{U}_{k-1})$, or the *belief state*—that is, for the purposes of optimal control, the belief state is known to be a *sufficient statistic*, essentially a lossless compression of the entire history of previously revealed information [2].

A POMDP model features properties that greatly simplify the computation of successive belief states. Firstly, because the (hidden) state takes only finitely-many values, say n, the distribution $P(x_k|\mathbb{Z}_k, \mathbb{U}_{k-1})$ is expressed by a length-n probability vector $b_k \in [0,1]^n$, whose components sum to unity with the ith component, $b_k[i]$, denoting the probability that state x_k is of its ith value. Secondly, by virtue of the Markov properties, implying in each stage k that (i) every state transition probability satisfies

$$p_{x_k|x_{k-1}}(u_{k-1}) = P(x_k|x_{k-1}, \mathbb{Z}_{k-1}, \mathbb{U}_{k-1})$$

and (ii) every sensor observation probability satisfies

$$q_{z_k|x_k}(u_{k-1}) = P(z_k|x_k, \mathbb{Z}_k, \mathbb{U}_{k-1}),$$

the sequence of belief states can be computed recursively [2]: given the preceding belief vector b_{k-1} and with the next control u_{k-1} and observation z_k both revealed, compute

$$\bar{b}[i] := q_{z_k|i}(u_{k-1}) \sum_{j=1}^{n} p_{i|j}(u_{k-1}) b_{k-1}[j] \qquad (7)$$

for every $i = 1, 2, \ldots n$ and then normalize the vector $\bar{b}$ i.e., then compute $b_k[i] := \bar{b}[i] / \left(\sum_{j=1}^{n} \bar{b}[j] \right)$ for each i.

The question that remains, of course, is how to design the policy by which each belief vector b_k maps into the control u_k. Decades of theoretical research has revealed interesting structure in the optimal policy, but also characterized the (worst-case) complexity of its computation to be prohibitive [12]. On the upside, a number of approximate solution algorithms have been developed that suitably exploit the problem structure and exhibit acceptable (average-case) complexity [13]. It is similarly more challenging, compared to the MDP counterpart, to evaluate the average-cost-per-stage performance of any given policy.

The analysis of our simplest POMDP model will proceed by using the iterative "witness" algorithm[3] [11] to approximate the optimal policy and then a simulation-based Monte-Carlo approximation to evaluate performance. The procedure must be repeated per instance of the POMDP model

[1] That c_F is absent from (1)-(3) stems from assuming perfect detectors.

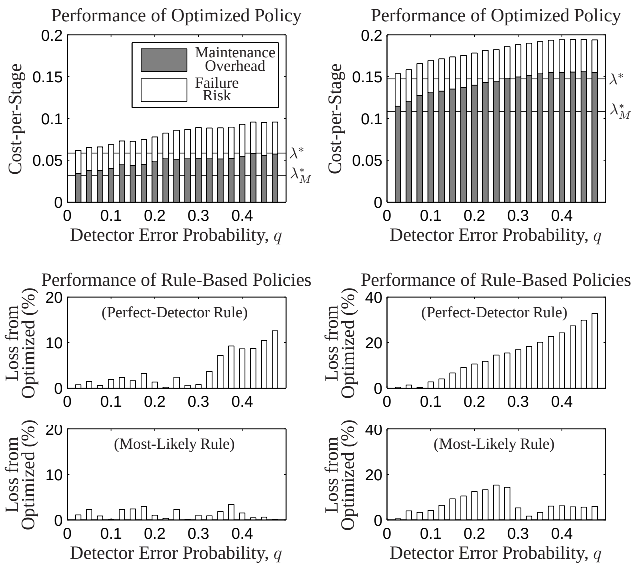
[2] More probability parameters could also capture (i) errors about the presence or absence of a security failure (e.g., see [8]) and (ii) controllable detectors or control-dependent sensor uncertainty (e.g., see [9]), but such extensions to the analysis of our model will not be pursued here.

[3] We utilize Cassandra's implementation (see http://www.pomdp.org).

i.e., per choice of values for parameters p_A, p_F, p_D, c_A, c_D, c_R, $q_{A|N}$ and $q_{N|A}$. (Recall that cost c_F is not pertinent as long as state F is assumed to be fully observable.) In our experiments, each call to the witness algorithm returns the policy $\tilde{\mu}$ obtained at the 500th iteration, taking the discount factor to be 0.9999 and the initial state distribution such that $x_0 = N$ with probability one. Each call to the Monte-Carlo performance evaluation returns the average-cost-per-stage over $T = 100,000$ simulated decision stages, each stage k realizing random variables x_k and z_k according to the given model, updating the belief vector b_k according to (7) and selecting control $u_k = \tilde{\mu}(b_k)$ according to the given policy.

Fig. 5 summarizes our experimental results for a particular set of costs and two different assumptions on attack dynamics, while taking the detector's two error probabilities to be equal and sweeping that value between perfect (at $q = 0$) and informationless (at $q = 0.5$). Consider first the upper plot in each column: the optimized performance is seen to degrade monotonically with increasing error probability q; more interesting is that against infrequent/slow attacks, this degradation manifests itself as increases in both maintenance overhead and failure risk whereas, against frequent/fast attacks, the degradation manifests itself almost exclusively as increased maintenance overhead. Also shown in these upper plots is the benchmark performance $\lambda^* = \lambda^D$ (from Subsection 3.1) in the perfect detector case.

The lower plots in Fig. 5 contrast the optimized performance against that achieved by two simpler rule-based policies. The *Perfect-Detector (PD)* rule ignores sensor uncertainty altogether, applying the "wait" or "defend" action depending on whether the detector reports "normal" or "attack", respectively; the *Most-Likely (ML)* rule updates the belief vector b_k via (7) but applies the "wait" or "defend" action depending on whether or not, respectively, probability $b_k[A]$ is less than probability $b_k[N]$. In the case of infrequent/slow attacks, even the PD rule is seen to perform close to optimum (e.g., less than 5% loss) for error probabilities up to $q = 0.3$, while the ML rule performs well for all q. In the case of frequent/fast attacks, however, the PD rule performs close to optimum only up to $q = 0.1$, exhibiting a peak loss of 35% at highest q; in contrast, the ML rule exhibits its peak loss of 15% at mid-range q, performing well in lowest & highest regimes of sensor uncertainty.



(a) Infrequent/Slow Attacks $(p_A, p_F, p_D) = (0.3, 0.1, 0.8)$ (b) Frequent/Fast Attacks $(p_A, p_F, p_D) = (0.7, 0.9, 0.8)$

Figure 5. In the case of imperfect detectors, the performance degradation of an optimized policy in comparison to that of two rule-based policies, taking $c_A = c_D = 0.1$, $c_R = 1$ and equal error probabilities $q_{A|N} = q_{N|A} \equiv q$.

References

[1] P. Veríssimo, N. Neves and M. Correia. "Intrusion-tolerant architectures: concepts and design," in *Architecting Dependable Systems*, LNCS 2677:3-36, Springer, 2003.

[2] D. Bertsekas. *Dynamic Programming and Optimal Control*, Athena Scientific, 1995.

[3] E. Jonsson and T. Olovsson. "A quantitative model of the security intrusion process based on attacker behavior," *IEEE Trans. on Software Engineering*, 23(4):1-11, Apr 1997.

[4] D. Nicol, W. Sanders and K. Trivedi. "Model-based evaluation: from dependability to security," *IEEE Trans. on Dependable and Secure Computing*, 1(1):48-65, Jan 2004.

[5] B. Madan, et al. "A method for modeling and quantifying the security attributes of intrusion tolerant systems," *Performance Evaluation*, 56:167-186, 2004.

[6] A. Årnes, et al. "Real-time risk assessment with network sensors and intrusion detection systems," in *Computational Intelligence and Security*, 388-397, Springer, 2005.

[7] Y. Huang, D. Arsenault and A. Sood. "Incorruptible self-cleansing intrusion tolerance and its application to DNS security," *J. of Networks*, 1(5):21-30, Sep 2006.

[8] K. Joshi, et al. "Automated recovery using bounded partially observable Markov decision processes," in *Proc. of Dependable Systems and Networks (DSN)*, 445:456, Jun 2006.

[9] O. Kreidl and T. Frazier. "Feedback control applied to survivability: a host-based autonomic defense system," *IEEE Trans. on Reliability*, 53(1):148-166, Mar 2004.

[10] S. Staniford, et al. "The top speed of flash worms," in *Proc. of ACM Workshop on Rapid Malcode*, 33-42, Oct 2004.

[11] L. Kaebling, M. Littman and A. Cassandra. "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, 101:99-134, 1998.

[12] C. Papadimitriou and J. Tsitsiklis. "The complexity of Markov decision processes," *Mathematics of Operations Research*, 12(3):441-450, Aug 1987.

[13] D. Aberdeen. "A (revised) survey of approximate methods for solving partially observable Markov decision processes," Technical Report, National ICT Australia, 2003.