Feedback Control Applied to Survivability: A Host-Based Autonomic Defense System

O. Patrick Kreidl, *Student Member, IEEE,* and Tiffany M. Frazier, *Member, IEEE*

***Abstract*—We address the problem of information system survivability, or dynamically preserving intended functionality & computational performance, in the face of malicious intrusive activity. A feedback control approach is proposed which enables tradeoffs between the failure cost of a compromised information system and the maintenance cost of ongoing defensive countermeasures. Online implementation features an inexpensive computation architecture consisting of a sensor-driven recursive estimator followed by an estimate-driven response selector. Offline design features a systematic empirical procedure utilizing a suite of mathematical modeling and numerical optimization tools. The engineering challenge is to generate domain models and decision strategies offline via tractable methods, while achieving online effectiveness. We illustrate the approach with experimentation results for a prototype autonomic defense system which protects its host, a Linux-based web-server, against an automated Internet worm attack. The overall approach applies to other types of computer attacks, network-level security and other domains which could benefit from automatic decision-making based on a sequence of sensor measurements.**

***Index Terms*—Computer security, empirical methods, intrusion tolerance, Markovian processes, numerical optimization, sensor uncertainty, stochastic control, survivable systems.**

ACRONYMS

ADS	Autonomic Defense System
ATCK	data tag for attack step
DFLT	data tag for default actuator command
IP	data tag for sensor report on network activity
KER	data tag for sensor report on kernel activity
KILL	data tag for kill-process actuator command
MDP	Markov Decision Process
NRML	data tag for web-server load
PID	data tag for sensor report on process activity
PO-MDP	Partially Observable Markov Decision Process
RCVR	data event tag for recovery actuator command

NOTATION

$S = \{1, 2, \ldots, n\}$	state space of cardinality n
$C = \{1, 2, \ldots, m\}$	control space of cardinality m
$Z = \{1, 2, \ldots, q\}$	observation space of cardinality q
k	decision stage index
$x_k \in S$	state of the process during stage k
$z_k \in Z$	observation received in stage k
$u_k \in C$	control applied in stage k
$I_k = (z_k, u_{k-1}, I_{k-1})$	observable information by stage k
$B_k = \Pr(x_k \mid I_k)$	probabilistic state at stage k
$\Pr(x_{k+1} \mid x_k, u_k)$	state transition probabilities
$\mathbf{F}$	system model, m $n \times n$ matrices
$\mathrm{c}(x_k, u_k, x_{k+1})$	state transition costs
$\mathbf{G}$	cost function, m $n \times n$ matrices
$\Pr(z_k \mid x_k, u_{k-1})$	sensor observation probabilities
$\mathbf{H}$	observation model, m $n \times q$ matrices
$B_k = \varphi(z_k, u_{k-1}, B_{k-1})$	estimation policy
$u_k = \mu(B_k)$	response policy
$\xi(\varphi \mid \mathbf{H}, \mathbf{F})$	average error per stage
$\lambda(\varphi, \mu \mid \mathbf{H}, \mathbf{F}, \mathbf{G})$	average cost per stage
$D \in \Re$	detector input measurement
$Y \in \{0, 1\}$	detector binary output
$Y = \delta(D)$	detector decision rule
$\gamma \in \Re$	detector binary threshold
$J(\gamma)$	detection expected cost
$\Pr(D \mid x_k)$	detection measurement distribution
$\Pr(x_k)$	detection prior distribution
$C(Y, x_k)$	detection cost function

Manuscript received February 27, 2002; revised April 24, 2002 and July 19, 2002. This work was supported by the Space and Naval Warfare Systems Center-San Diego under Contract N66001–00-C-8030. Responsible Editor: N. Ye.

O. P. Kreidl is with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: opkreidl@alum.mit.edu).

T. M. Frazier is with ALPHATECH, Inc., Arlington, VA 22203 USA (e-mail: tiffany.frazier@alphatech.com).

Digital Object Identifier 10.1109/TR.2004.824833

I. INTRODUCTION

THE increasing reliance on information systems within critical military and civilian operations coupled with the proliferation of malicious intrusive activity [1] motivates continued improvements in computer security. Malicious intrusions can be described at many levels, including in terms of information system impact (e.g., denial of service, theft of service, etc [2]) or in terms of attacker objectives (e.g., defacing a government web server in order to make a political statement). An intelligent attacker can automate the execution of successive intrusion attempts, perhaps even mutating its key properties before each attempt to evade or bypass static security mechanisms, and potentially compromise the information system's ability to provide essential services [3].

The desirable capability of an information system to dynamically preserve its essential functionality and computational performance in the presence of security intrusions is defined as *survivability* [4]–[6]. A survivability objective, especially when considering the dynamic & uncertain nature of security failures due to malicious intrusions, mandates that offline techniques employed during design be supplemented by online techniques to be employed during operation [7]–[9]. Proven design paradigms for fault-tolerant and safety-critical systems also suggest survivability can be achieved through a sequence of system partitioning, subsystem design, and system-wide integration [10]. For example, improved survivability of each single computer within a networked information system can buy time for upper level components of a multi-layer security architecture to react with more coordinated diagnosis and counter-attack strategies, ultimately enabling improved global security.

Most automated security-related responses built upon real-time intrusion detectors [11]–[15] have been ad-hoc, typically relying on just individual reports at a point in time. However, intrusion detection is a relatively new & immature area [16], [17]; and the frequency of false positives leads to a high potential for inappropriate ad-hoc response. In addition to false positives, there can be false negatives which analogously confound ad-hoc response selection. In principle, security-related actuators are not limited to responses to thwart immediate attack objectives, but may also include responses related to sensor management, adjustments to system functionality as well as responses related to security posture or recovery. Especially when considering multi-stage intrusions, it is possible that certain preemptive responses may suppress or interfere with subsequent sensor reports. From these perspectives, prudent response selection needs to anticipate the outcome of initiated actuators with regard to both immediate & future attack detection & defense.

Modern control theory provides a well-established mathematical framework for capturing competing decision objectives over an extended period of time within a dynamic & uncertain environment. The solution to a control-theoretic problem formulation distinctly recognizes the opportunity of *information feedback* during online operation; that is, the form of the optimized decision strategies allow for each online decision to directly depend on the most recent information available from all observable data. The use of information feedback can dynamically merge and correlate multiple sensor streams over time, yielding better decision-making than would be achievable by considering each sensor stream in isolation. Feedback also allows the modeled effectiveness and usage history of multiple response actuators to factor into subsequent decision-making.

This paper focuses on the application of feedback control theory to improve survivability for a single *host* computer, emphasizing the real-time sensing of locally apparent intrusive activity with automated response selection. In particular, we leverage control-theoretic architectures, mathematical models and numerical algorithms [18]–[30] to prototype in software a real-time, host-based *Autonomic Defense System* (ADS). Fig. 1 illustrates the general ADS architecture, integrating:

i) the *information system* to be protected,

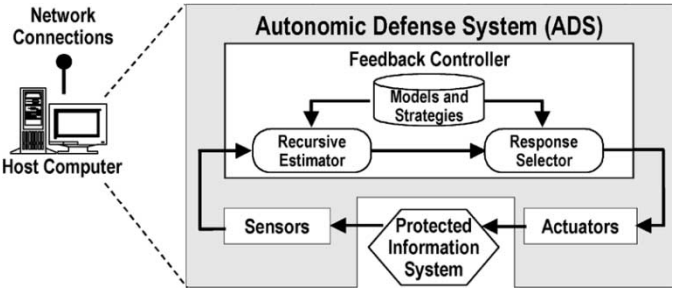

Fig. 1. Host-based autonomic defense system integrates a model-based feedback controller with security-related sensors and actuators to dynamically defend against security intrusions.

ii) a set of *sensors* which repeatedly report on the presence or absence of normal or intrusive activity,
iii) a set of *actuators* which implement security-related defensive responses, and
iv) a *controller* which orchestrates all available sensor and actuator assets to maximize host survivability.

The controller is decomposed into a *recursive estimator* and a *response selector*, where the recursive estimator processes a stream of sensor observations, repeatedly updating its state estimate, to drive the response selector which dynamically chooses from available defensive actuators.

Feedback control refers to the process of repeatedly:

- receiving a new sensor observation,
- estimating its implication within the context established from past observations & responses, and
- using that estimate to drive the selection of a new response.

Even though the nature of an intrusive attack, the implications of a sensor observation, or the outcome of an actuator response, may not be fully predictable; all are anticipated using statistical models before each control decision is made. The survivability objective is expressed as the minimization of a certain mathematical cost that quantifies a tradeoff between an overall "maintenance cost" associated with defensive response, reducing the probability of an attack's completion, versus an overall "failure cost" associated with an attack's completion. The feedback controller can relate, by way of estimation and response strategies optimized with respect to expected total cost, any specific sequence of sensory input to a specific sequence of commanded output which alters the future evolution of the information system's security status in desired ways.

The remainder of the paper is organized as follows. Section II characterizes the host-based survivability problem as a sequential decision process under uncertainty so that well-established control-theoretic formulations can be applied. A particularly well-studied feedback control problem, called a stationary Partially-Observable Markov Decision Process (PO-MDP), is identified as the basis to develop a prototype ADS. Section III describes the experimental setup of a specific security scenario, to protect a Linux-based web server from automated Internet worm attacks, where sensors are provided by a commercially available intrusion detection package. We describe the main features of the emulated, threatened web server environment, the design & integration of the host-based security assets, as well as the data collection process. Section IV illustrates, for this practical

setting, the key model development steps and concludes with encouraging results while operating the prototype ADS within the laboratory web server environment.

II. Methodology

This section first discusses the rationale for applying feedback control theory to the computer security domain. We state the main constructs of general feedback control problems, qualitatively highlighting important aspects of the supporting mathematical models and optimization algorithms. The interested reader is referred to [18]–[30] for specific mathematical details and supporting formal proofs. A basic and well-studied feedback control problem, called a stationary Partially-Observable Markov Decision Process (PO-MDP), is precisely described because it serves as the specific model and algorithm leveraged in Sections III and IV.

A. Technical Rationale

General control theory addresses the design of decision strategies to minimize a cost function, or maximize a reward function, as the state of a dynamic environment evolves over an extended period of time. In the presence of uncertainty, either in the way the state evolves or in the way the state is observed or both, it is not a well-posed problem to optimize this criterion directly. Rather, stochastic control formulations optimize an expected value of the cost, resulting in control strategies that best satisfy the survivability objective in an on-average sense. With inherent uncertainty of the computer security domain in mind, casting the survivability objective to meet probabilistic guarantees is a more realistic goal than requiring deterministic guarantees, and making severe assumptions to achieve them. Stochastic control has been applied to safeguard operation and optimize the performance of a wide variety of systems, ranging from decision-making for economic and social systems, to trajectory control for robots & vehicles.

For both signature & anomaly detectors, the design challenge involves the definition of a suitable model, either a signature or a profile, and the development of an underlying pattern recognition algorithm which monitors ongoing activity and can measure a *distance* from that model. During operation, the detector throws its alert when ongoing activity produces a distance exceeding a specified *threshold*, or a maximum tolerance on the magnitude of that distance, which introduces the problem of selecting the threshold value. Choosing too large a threshold results in a detector that potentially fails to yield alerts when the activity of interest is truly present, an event commonly referred to as a *missed detection* or *false negative*; choosing too low a threshold results in a detector that potentially sounds alerts when the activity of interest is truly absent, an event commonly referred to as a *false alarm* or *false positive*. There is also the operational challenge of detector *calibration*, or training the supporting model and pattern recognition algorithm within the target information system to be monitored. Naturally, the reliability of the estimated state that is based on these alerts is strongly dependent on the adequacy of this calibration, something acknowledged as difficult & tedious to quantify in practice [16], [31], [32].

With regard to ad-hoc response, frequent missed detections increase the potential for costly security failures while frequent false alarms increase the potential for costly over-application of defensive countermeasures. The inherent degree of uncertainty associated with the output from either category of detectors motivates a statistical characterization of intrusion sensors [33], [34]. Such statistical methods rely on systematic procedures to select the distance thresholds and experimentally quantify rates of missed detection and false alarm [20]–[22]. Given the statistical accuracy of a detector, a sequence of measurements can be processed to infer context for response selection which can be more reliable than reacting to any single measurement in isolation. However, characterizing a software-based detector is complicated by the fact that, being deeply instrumented into the target information system, the act of sensing itself affects overall operation. Thus, any attempt to benchmark the detector's statistical accuracy prior to integration into the target information system may yield a characterization which does not match with the operational detector [32], significantly confounding the ability to reliably infer response context from the measurements.

Even accurately modeled sensor & actuator uncertainty results in the possibility of applying inappropriate response, by either selecting defensive action under nonintrusive activity, or inaction during intrusive activity. There is an inherent survivability tradeoff: inappropriate action draws from the host's total resources whereas inappropriate inaction permits attack progression. A stochastic control formulation accepts that all decisions must be made based on imperfect information from sensors and imperfect effectiveness from actuators and, furthermore, recognizes the inherent tradeoff between an expected "maintenance" cost due to action versus an expected "failure" cost due to inaction.

The application of control theory to the computer security problem domain has been proposed several times [34]–[37], each emphasizing different aspects of the broadly applicable mathematical framework. In [34], and [35], motivated by an analogy between intrusion detection and statistical process control, the emphasis is on architecture & design as well as the statistical characterization of intrusion detectors via a systematic empirical procedure using experimentation data. [36] also discusses architecture and recognizes the inherent sensor uncertainty, arguing a role for signal processing methods to amplify the attack signal from normal noise within a sequence of measurements. However, [34]–[36] do not address the impact of architecture and sensor uncertainty with emphasis on automated response to meet a survivability objective. Finally, [37] advocates a full feedback loop in its approach to circumvent the limitations of rule-based responses and also addresses the value of simulation-based strategy optimization to synthesize the controller. However, the underlying problem formulation does not appear to explicitly account for sensor uncertainty, and pertinent details about the supporting models are omitted, including how to develop them in either a systematic or empirical manner. While perhaps not treating any single issue to the extent of [34]–[37], our approach touches on all of them in a comprehensive mathematical framework, providing insight into how each issue impacts another when the ultimate objective is real-time, host-based autonomic defense.

B. Feedback Control Overview

A typical feedback control problem formulation consists of three key mathematical models. Firstly, a multi-stage *system model* characterizes the stochastic evolution of the *state* and its dependence on *control*. Secondly, a *cost function* characterizes the relative undesirability of all possible controls and single-stage outcomes defined by the system model. Thirdly, unless we are assuming the controller has perfect access to the true state, a multi-stage *observation model* characterizes the control-dependent statistical correlation of each *observation*, or processed measurement, with the true state. Here, *state* is defined according to Markovian assumptions so that knowledge of its current value implies any additional information about the past is redundant or irrelevant for choosing future controls. A *control* is defined as any response available to the controller with potential to influence the outcome of either present or future decision stages. *Active* controls refer to responses that may directly alter the true state, while *passive* controls only influence how observations are made in the future. Of course, a control may potentially influence both future states & future observations, being simultaneously active & passive.

The concepts of controllability, observability, and stability are of paramount importance in control theory. Qualitatively, *controllability* refers to a measure of how effective the controls are with respect to altering the true state. If a state is uncontrollable, applying control has no predictable influence on how the future evolves from that state. Complete controllability, on the other hand, implies it is possible through some finite sequence of actions to alter the true state to any other desired state. Similarly, *observability* refers to a measure of how informative the sensor observations are with respect to estimating the true state. If a state is unobservable, the sensor observations have no useful information content to help infer that true state. Complete observability, on the other hand, implies it is possible through some finite sequence of observations to uniquely determine the true state. Finally, *stability* refers to a system which, in the absence of external input, eventually settles to some equilibrium state. When stability of the controlled system is not satisfied, the immediate objective must be to stabilize the system and, only upon stabilization, is it possible to proceed with objectives to improve controller performance. Provided the system model is at least a partially controllable characterization, the observation model is at least a partially observable characterization, and the stability guarantees are preserved, then the cost function allows the performance problem to be stated in a manner well-suited for mathematical optimization tools.

In a dynamic and uncertain multi-stage process, a single-stage decision cannot be viewed in isolation. Rather, the desire for low immediate cost must be balanced against the undesirability of high expected future costs. Numerical optimization via *dynamic programming* mathematically captures this tradeoff, explicitly accounting for all modeled uncertainty in the process. The technique supports a general problem formulation and, conceptually, characterizes a numerical algorithm which terminates with a minimum-cost, or *optimal*, solution. The solution to a dynamic program is computed offline but, when applied online, is not just a specific sequence of controls. Rather, the solution is *a control strategy* which explicitly recognizes the sequential nature of the problem, and exploits *information feedback*; allowing each decision stage to depend on information gathered over previous stages. While the exact dynamic programming algorithm is computationally prohibitive when scaled to most practical problems, the mathematical framework retains a practical value via the insight it provides to determine an appropriate balance between problem representation & solution computation, and, ultimately, guide the development of tractable yet effective *sub-optimal* solutions.

Conceptually, dynamic programming algorithms treat *imperfect state information* problems no differently from *perfect state information* problems. Instead of a control strategy relying on the true state which summarizes all essential context, there is a "state of information" which summarizes all observable context consisting, at best, of all received observations and all past controls. With imperfect state information, each stage of the on-line control strategy is commonly decomposed into two steps involving a *statistical estimate* of the true state: an *estimation policy* that recursively generates the current estimate as a function of the previous estimate, preceding control, and the current observation; followed by a *response policy* that selects the current control as a function of the resulting estimate. Ideally, each recursive estimate is equivalent in information content to directly feeding the controller with all observable data up to the control decision, a property defined as *sufficient*. More typically, the decomposition into a *recursive estimator* and a *response selector* is a sub-optimal solution exercised in the interest of computational tractability.

Markov Decision Process (MDP) models describe a particular class of multi-stage feedback control problems which have been studied extensively and repeatedly applied within, for example, the realm of operations research, economics, and computer & communication networks. Dynamic programming techniques are especially well-developed for stationary MDP models with finite state and control spaces. A finite-state, finite-control MDP model applies when, at each decision stage, response selection from a finite set of possible actions depends on only a finite set of essential considerations. Under certain technical conditions, the dynamic programming algorithm can be reduced to a special system of nonlinear equations, enabling specialized algorithms which guarantee convergence to the optimal solution, and do so with significantly greater computational efficiency than for general dynamic programs.

A *Partially-Observable* (PO) MDP model assumes imperfect state information where the controller receives at each stage one of a finite set of possible sensor observations. A finite-state, finite-control, finite-observation PO-MDP can, under certain technical conditions, be reformulated to an equivalent MDP involving the same finite control-space but an infinite state space corresponding to all possible values of the *probabilistic state*, or a sufficient statistic having the intuitive form of a probability distribution on the original finite state-space. Moreover, the recursive estimation policy which generates successive probabilistic state estimates has a closed-form optimal solution. While the dynamic programming solution no longer reduces to the same computationally efficient form as for fully-observable MDPs, the optimal response policy of a

PO-MDP has an intuitively appealing structure for the purposes of approximation and implementation by computer.

C. Stationary PO-MDP Formulation

A stationary MDP involving a finite set of states and a finite set of controls is structurally characterized by a state space S of n distinct states, or $S = \{1, 2, \ldots, n\}$, and a control space C of m distinct controls, or $C = \{1, 2, \ldots, m\}$. Consider a sequence of decision stages where each stage k begins in one of the n states, $x_k \in S$. The controller then selects one of the m controls, $u_k \in C$, upon which a transition to the next state, $x_{k+1} \in S$, occurs according to $m{\cdot}n{\cdot}n = mn^2$ known state transition probabilities $\Pr(x_{k+1}|x_k, u_k)$. The state transition event simultaneously ends the kth decision stage and begins the next. The conditional dependence of the transition probabilities on the current state x_k and the selected control u_k reflects the extent to which anticipated dynamics and the outcomes of the active controls must be modeled. The relative undesirability of each of the mn^2 possible single-stage outcomes, involving the current state x_k, applied control u_k and next state x_{k+1}, are quantified by state transition costs $c(x_k, u_k, x_{k+1})$. Note that the arguments of the transition costs, (x_k, u_k, x_{k+1}), are identical to the arguments that define each possible single-stage outcome in the system model.

We assume the controller cannot access the exact state x_k but instead receives one sensor observation z_k prior to selecting each control u_k. A finite-observation PO-MDP is structurally characterized by an observation space Z of q distinct sensor observations, or $Z = \{1, 2, \ldots, q\}$, in addition to a state space S and a control space C. Each observation $z_k \in Z$ occurs according to $m \cdot n \cdot q$ known sensor observation probabilities $\Pr(z_k|x_k, u_{k-1})$. The conditional dependence of the observation probabilities on the current state x_k and the previously selected control u_{k-1} reflects the extent to which the anticipated implications of a sensor observation and the outcomes of the passive controls must be modeled.

All the parameters of a stationary finite-state, finite-control, finite-observation PO-MDP model can be conveniently organized into a family of control-dependent matrices: m $n \times n$ transition probability matrices $\mathbf{F}$, m $n \times n$ transition cost matrices $\mathbf{G}$ and m $n \times q$ observation probability matrices $\mathbf{H}$. More precisely, for each i, $j \in S$ & $a \in C$, let the given transition probability $\Pr(x_{k+1} = j|x_k = i, u_k = a)$ correspond to the element in the ith row and jth column of matrix $\mathbf{F}(a)$. Similarly, let the given transition costs $c(x_k = i, u_k = a, x_{k+1} = j)$ populate the matrices $\mathbf{G}(a)$. Finally, for each $i \in S$, $a \in C$ & $o \in Z$, let the given observation probability $\Pr(z_k = o|x_k = i, u_{k-1} = a)$ correspond to the ith row and oth column of matrix $\mathbf{H}(a)$.

The above stationary model may utilize the infinite-horizon assumption so that the optimal control strategy is also stationary. The above model also satisfies all technical assumptions required to utilize the probabilistic state estimate as a sufficient statistic and optimally decompose the feedback controller into a recursive estimator followed by a response selector. Specifically, let I_k denote all observable information received prior to selecting the kth control u_k. Let B_k denote the stage k estimate having the form of a column vector of length n whose

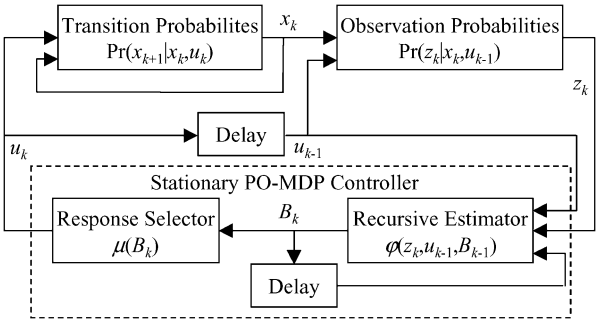

Fig. 2. Decomposition of the stationary PO-MDP controller and the associated online data flows captured by the transition and observation probabilities.

ith element is $\Pr(x_k = i|I_k)$. Note that each probabilistic state estimate is an element from the uncountable space, $B_k \in \{\mathbf{v} \in [0,1]^n \,|\sum_{x=1}^{n} [\mathbf{v}]_x = 1.0\}$ where $[\mathbf{v}]_i$ denotes the ith component of a vector $\mathbf{v}$. Intuitively, each ith element of B_k represents the relative confidence that state i is indeed the true state in stage k. Thus, if state i is without a doubt the true state, B_k is a unit vector in the ith component; conversely, when no meaningful information has been provided to the estimator, B_k might be the uniform distribution over the n states.

Based on the probabilistic state estimate, each stage of the on-line control strategy is implemented via two straightforward function evaluations: an *estimation policy* φ that outputs each estimate B_k according to a recursion

$$B_k = \varphi(z_k, u_{k-1}, B_{k-1}) \tag{1}$$

where z_k denotes the current observation, u_{k-1} the preceding control, and B_{k-1} the preceding estimate;, and a *response policy* μ that outputs the selected control u_k according to

$$u_k = \mu(B_k). \tag{2}$$

Fig. 2 depicts the controller decomposition, the associated on-line data flows characterizing each decision stage, and the role of the system and observation models. Note that the cost function influences the optimization which yields desirable response policies.

Consider a specific stationary PO-MDP model characterized by the family of matrices $\mathbf{F}$, $\mathbf{G}$ and $\mathbf{H}$. Let $d(B_k, x_k)$ denote a given distance metric between the statistic B_k and the true state x_k. For the probabilistic state estimate, a valid distance metric is

$$d(B_k, x_k) = \frac{1}{\sqrt{2}} \|B_k - \mathbf{1}_n(x_k)\| \tag{3}$$

where $\|\bullet\|$ denotes the Euclidean norm and $\mathbf{1}_\ell(i)$ denotes the ith unit vector in an ℓ-dimensional vector space. Dividing by the factor $\sqrt{2}$ normalizes the error distance to units of probability. Then, for any specific estimation policy φ of the form in (1), the *average error per stage* is defined by

$$\begin{aligned}\xi(\varphi|\mathbf{H}, \mathbf{F}) &= \lim_{K\to\infty} \frac{\xi_K(\varphi|\mathbf{H}, \mathbf{F})}{K} \\ &= \lim_{K\to\infty} \frac{1}{K} E\left[\sum_{k=0}^{K-1} d(B_k, x_k) \middle| \mathbf{H}, \varphi, \mathbf{F}\right] \end{aligned} \tag{4}$$

where the expectation $E[\bullet]$ is over all random variables implied by the probabilities $\mathbf{F}$ and $\mathbf{H}$, as well as the particular estimation policy φ. Also given any specific response policy μ of the form in (2), the *average cost per stage* is defined by

$$\lambda(\varphi,\mu|\mathbf{H},\mathbf{F},\mathbf{G}) = \lim_{K\to\infty}\frac{\lambda_K(\varphi,\mu|\mathbf{H},\mathbf{F},\mathbf{G})}{K} = \lim_{K\to\infty}\frac{1}{K}E\left[\sum_{k=0}^{K-1}c\left(x_k,\mu(B_k),x_{k+1}\right)\middle|\mathbf{H},\varphi,\mathbf{F}\right] \tag{5}$$

where the expectation $E[\bullet]$ is over all random variables implied by the probabilities $\mathbf{F}$ and $\mathbf{H}$, as well as the particular estimation policy φ.

In general, the overall control objective is to minimize λ in (5) with the simultaneous design of estimation and response policies, φ and μ. When the statistical estimate is known to be sufficient, the overall control objective can be decomposed into isolated estimation and response objectives without loss of optimality. The overall estimation objective is to minimize ξ in (4) with the design of estimation policy φ. For the stationary PO-MDP model, probabilistic analysis yields the optimal estimation policy, denoted by φ^*, via the closed-form recursion

$$\varphi^*(z_k=o,u_{k-1}=a,B_{k-1}) = \frac{[\mathbf{H}(a)]_o * [\mathbf{F}(a)' B_{k-1}]}{[\mathbf{H}(a)]_o' [\mathbf{F}(a)' B_{k-1}]} \tag{6}$$

where

- $\mathbf{A}'$ denotes the transpose of a matrix $\mathbf{A}$,
- $[\mathbf{A}]_o$ is the column vector corresponding to the oth column of matrix $\mathbf{A}$,
- $[\mathbf{v}]_i$ is the ith coordinate of a vector $\mathbf{v}$, and
- $\mathbf{v_1}*\mathbf{v_2}$ is the vector whose ith coordinate is the scalar $[\mathbf{v_1}]_i[\mathbf{v_2}]_i$.

Once the estimation policy φ is fixed, the overall response objective is to minimize λ in (5) with the design of response policy μ. The optimal stationary response policy, denoted by μ^*, can be parameterized by a straightforward matrix multiplication with the length-n probabilistic state vector,

$$\mu^*(B_k) = \arg\min_{u_k\in C}[\mathbf{Q}^*(B_k)B_k], \tag{7}$$

where the minimum element of the resulting length-m vector maps to the optimal control. Here, $\mathbf{Q}^*$ is a function to be designed, mapping the probabilistic state space into a set of $m\times n$ matrices. For a given state estimate B_k, the aith element of matrix $\mathbf{Q}^*(B_k)$ corresponds to the average cost per stage if state $x_k=i$ is in fact true, control $u_k=a$ gets selected, and optimal control continues for all future decision stages. While exact determination of $\mathbf{Q}^*$ is typically not possible, (7) provides a useful parameterization from which approximation methods can be engineered offline. Online operation then consists of a "look-up" to obtain the matrix whose multiplication with the realized probabilistic state vector, itself recursively generated through elementary matrix calculations via (6), yields optimized response selections via the operation expressed in (7).

D. Practical Remarks

Strict stability for a finite-state, finite-control stationary MDP model is not of concern. Under broadly technical assumptions, the probability of occurrence of each modeled single-stage outcome (x_k,u_k,x_{k+1}) tends to a steady-state distribution which depends exclusively on the system model $\mathbf{F}$, the observation model $\mathbf{H}$, and fixed stationary policies φ & μ. There can be periodic behavior, yielding distinct steady-state solutions for each stage of the period, but the MDP is still technically considered a stable process.

Together, the controllability & observability properties of the modeled PO-MDP correspond directly to the achievable performance of a control strategy. Once the controllability & observability properties are deemed acceptable, the following technical assumptions must be satisfied for the simplest recursive state estimator φ^* & the response selector μ^* defined by (6) & (7), respectively, to apply exactly:

- The stationary model applies over an infinite number of stages, implying the control strategy is stationary.
- Conditioned on the true state and previous control, the observation distribution at each stage is s-independent from all information related to past decision stages.
- Sensor measurement and actuator response delays are negligible compared to the typical duration of each decision stage.

In practice, the first assumption is never strictly satisfied but (6) & (7) may still perform well when the process parameters vary slowly over a large number of stages.

The second assumption implies the recursive estimates generated by (6) are sufficient, allowing the feedback controller to be decomposed into a recursive estimator followed by a response selector without loss of optimality. The assumption is usually not satisfied in practice; for example, a single sensor stream may be correlated with itself across time and multiple sensor streams may be cross-correlated. Regardless, the controller decomposition is typically exercised in the interest of tractability. Utilizing correlation filters when processing incoming measurements is one way to yield an observation stream which better satisfies the assumption, but introduces additional computational expense to every stage of online operation.

The third assumption allows the simple matrix-based models & computations defined by (1) through (7) to suffice in generating estimates and selecting controls. If delays are significant, the system model has to be generalized to include a temporal-dependence on the transition probabilities used in each decision stage. Unfortunately, as for handling correlated sensors, handling delays adds computational expense to each stage of online operation.

III. Experimentation Scenario

This section describes the design of a prototype host-based ADS intended to protect a Linux-based web server from automated Internet worm attacks. Automatic response is desired because the attack phases of

i) vulnerability identification and exploitation,
ii) local damage infliction, and

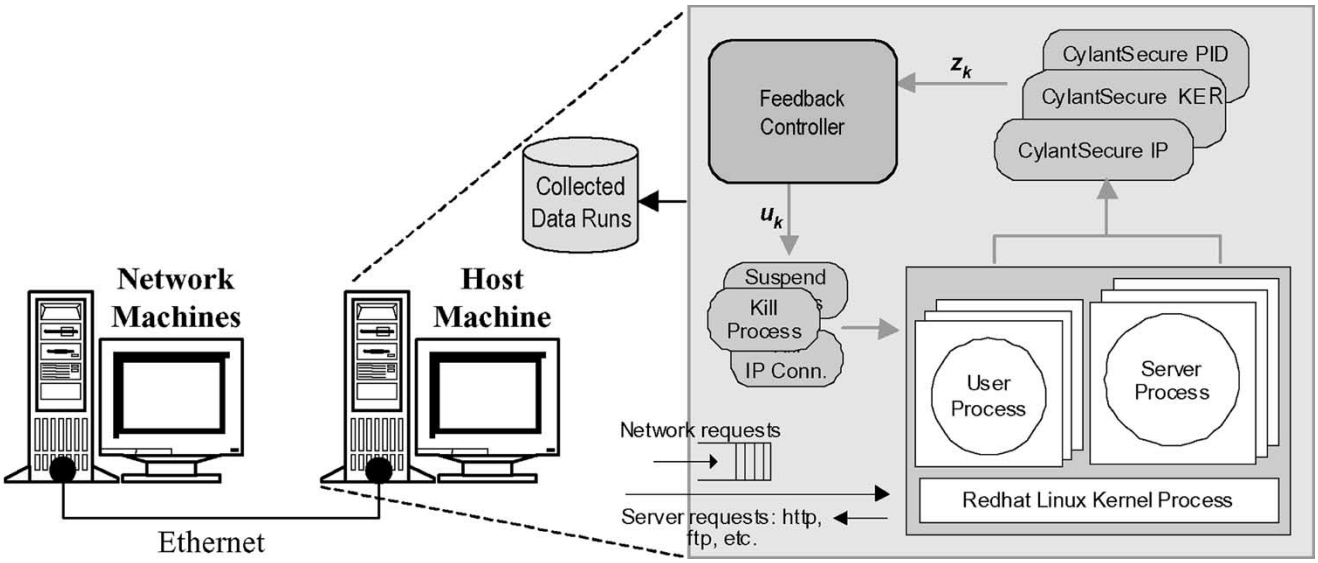

Fig. 3. Basic laboratory setup for the experimentation scenario considering a web-server under the threat of an automated Internet-worm attack.

iii) subsequent transformation of the host into an attacker evolve at machine speed, making it impractical for manual intervention to detect the attack and prevent the worm from spreading.

The relevance of this scenario is supported by [38], where recent statistics reflect both the growing numbers of organizations experiencing web-server attacks and the proliferation of Internet worms. Fig. 3 illustrates the overall laboratory setup, and the basic integration architecture for all of the security assets composing the prototype ADS; namely the sensors, the actuators, and the model-based feedback controller. The following subsections provide details.

A. Laboratory Setup

The goal of the laboratory setup is to emulate a security scenario involving a vulnerable web server. Several Dell Optiplex GX110 PCs, each running Red Hat Linux v6.2 with a 733 MHz Pentium III processor & 256 Mb of memory, are networked by 100 Mb/sec Ethernet connections. During experimentation, one machine is designated as the protected web server and the others as clients. The clients generate normal web traffic via random page requests to the server. On the server, normal user session and cgi-script activity is generated through a number of Bourne Shell, Perl, & Expect scripts executing basic Unix commands. Both the web traffic, and the local user sessions on the server can be set to "low," "medium," and "high" activity levels.

A client can also, at any time, initiate an Internet worm attack. The particular worm we use as an exemplar is a variation on the Ramen worm [39], a cataloged attack which exploits vulnerabilities present in relatively recent Linux distributions. Though the experiments involve a specific Internet worm, the goal is to develop the prototype ADS to recognize a broader class of Internet worm attacks. Table I outlines an Internet worm's generic steps grouped into separate phases according to the sub-objectives of the attack sequence [2]. The typical characteristics of an Internet worm attack include:

- port scanning and probing to identify potential victim systems & specific vulnerabilities present on those systems,
- user-to-root access exploitation,
- downloading and installing an attack payload, and,
- ultimately, conversion of the victim into a new attacker.

Some candidate countermeasures are common to each of these steps as well, though specifics, e.g., which attack process to kill, may differ significantly.

We have instrumented the attack to utilize one of three root shell exploits (**wu-ftpd**, **rpc.statd**, & **named**), and one of two port scanners (**nmap** & **synscan**). This instrumentation provides essentially six main variations of a malicious worm with which to attack the victim. We can also insert random delays into the instrumented attack to vary the timing between key attack steps. Variations on the attack are useful to determine the extent to which the prototype ADS, whose underlying feedback controller is trained using a particular instance of the worm attack, can defend against attacks which are distinct from those used in the training set. Finally, to facilitate batch data collection, the attack instrumentation also includes a "clean-up" script which undoes attack damage, and reinstates the security of all machines, as well as restarts normal operation across the laboratory network.

B. Security Assets

The sensors on the host machine are software-based anomaly detectors included as part of the commercially-available CylantSecure system [15] developed by Software Systems International [40]. CylantSecure consists of:

- a specially instrumented Linux kernel,
- a normal system behavior database,
- a separate process called "Watcher," and
- (optionally) a "Console" that is the security manager's interface to CylantSecure.

The instrumented Linux kernel contains software probes, which continuously collect data about the behavior of the running system. These behavior data are divided into three separate categories:

- one for probes associated with process activity (PID),
- one for network activity (IP), and
- the remainder for general kernel activity (KER) for what remains.

TABLE I
KEY PROPERTIES OF AN AUTOMATED INTERNET WORM ATTACK

Step	Attack Objectives	Candidate Countermeasures
	Phase 1: Gain privileged access to victim's configuration	
1	Scan network and probe each computer in search of vulnerable victim	heighten awareness, scrutinize port activity, ignore suspicious source IP
2	Bypass security of an identified victim by exploiting the vulnerability	block entrance port, sleep/kill exploited process or service
	Phase 2: Inflict local damage on victim	
3	Download the payload onto the victim containing programs that propagate the attack	sleep/kill shell connection, sleep/kill download, block disk access
4	Install the downloaded programs to be executable by the victim	sleep/kill shell connection, delete payload, sleep/kill installation utility, block disk access
	Phase 3: Convert victim to an attacker	
5	Execute the installed programs that propagate the worm	halt operations and alert administrator to initiate recovery sequence

During online operation, these data streams are sent to the "Watcher" process which compares each stream of incoming execution profiles with its normal profile (generated during a *calibration phase*) to produce numerical distance measurements quantifying the measured level of anomalous kernel behavior. Note that all three streams provide a distance value per report, but the PID stream also includes context about the associated process and the IP stream includes context about the associated network connection. The rate of sensor reports during normal operation is directly dependent on the amount of kernel activity, which during our experimentation ranged from one report every 10 to 1000 milliseconds on-average under the varying loading conditions.

We consider three different actuators, corresponding to the minimal set of responses necessary to potentially counter or recover from a generic worm attack. The first actuator corresponds to the default response and is purely passive, meaning it influences only the way a future observation is generated but offers no potential to modify the evolution of an actual attack. An actuator that can kill a process has potential to counter the worm during its first two phases, providing the correct process is identified and killed in time. Targeting a nonattack process degrades quality-of-service and is assumed undesirable. By the time the worm has entered its third and final phase, attack objectives are no longer effectively countered by simply killing processes, and a more elaborate recovery mechanism is required. This recovery mechanism is assumed to severely disrupt the normal operation of the host, so is considered an appropriate response only after the attack enters its third phase. Ideally, the less drastic kill-process actuator is appropriately applied in the earlier phases so that the recover action need never be utilized. Denote the default, kill-process, and recovery actuator by DFLT, KILL, and RCVR, respectively.

Note that the feedback controller, relying on a PO-MDP model & policy as described in Section II to perform the recursive estimation & response selection calculations, is implemented as a separate software process on the host machine. It serially receives, via a shared-memory interface, each measurement generated by CylantSecure. To fit the PO-MDP formulation, a mapping from this time-stamped sequence of real-valued distances to a discrete-time sequence of single-stage observations must be specified. The default actuator, denoted by DFLT, represents this repeated activity of processing incoming measurements for the next decision stage. It is, in effect, a "wait-and-see" action parameterized by a *staging scheme*, or rule by which the next decision stage is generated; and a *detection scheme*, or rule by which multiple real-valued measurements received within a stage get mapped to a finite observation space.

For simplicity, we restrict ourselves to a staging & detection scheme which independently considers each of the three data streams, KER, PID, & IP, in a similar manner. Let the staging scheme per stream ℓ be parameterized by an interval of time between estimator updates, or the *update period* τ_ℓ, coupled with how all distance values received in a single period are reduced to a single value, or the *aggregation rule* ρ_ℓ. Fixing a staging scheme (τ_ℓ, ρ_ℓ), we can rely on Bayesian methods to optimize a detection scheme. The general form of the optimal detector is a *decision rule* δ which partitions the infinite measurement space into r disjoint decision regions, each region corresponding to one of a finite number r of possible output observations.

Again for simplicity, we assume a binary detector output, or $r = 2$, per measurement stream ℓ, with each stream's decision rule parameterized by a single threshold value γ_ℓ. That is, given a single-stage (perhaps aggregated) measurement $D_0 \in \Re$ on stream ℓ, we consider decision rules $\delta_\ell : \Re \rightarrow \{0, 1\}$ of the form

$$\delta_\ell(D_0) = \begin{cases} 0, & D_0 \leq \gamma_\ell \\ 1, & D_0 > \gamma_\ell \end{cases} \tag{8}$$

where output "0" denotes the *safe* observation and "1" the *alert* observation. Note that the true state from which any one measurement is produced is one of the n components of the state space S. Viewing any D_0 as a realization of random variable D, optimal threshold selection depends on:

i) n *measurement distributions* $\Pr(D|x_k)$ which can be extracted from collected data,
ii) a *prior probability distribution* $\Pr(x_k)$, obtained via a combination of assumptions and data, and

iii) a *detector cost function* $C : \{0,1\} \times S \rightarrow \Re$, where $C(Y_0, i)$ represents the relative undesirability associated with detector output $Y_0 \in \{0,1\}$ when $i \in S$ is the true state.

For any fixed threshold γ_ℓ, its *expected detection cost* is then defined by (9), as shown at the bottom of the page, and thus, within the specific class of binary decision rules described by (8), the optimal threshold γ_ℓ^* minimizes (9). So, given collected data as well as the prior probability distributions and detector cost functions, applying (8) & (9) reduces the detection scheme parameterization to consist of three binary thresholds $(\gamma^*_{\rm KER}, \gamma^*_{\rm PID}, \gamma^*_{\rm IP})$, one per CylantSecure data stream.

The kill-process actuator, KILL, extends the DFLT parameterization with a parameter which also specifies the *targeting scheme* $\sigma_{\rm PID}$ to be employed, based on the context provided by the CylantSecure PID stream. In other words, the KILL actuator maintains a table of processes associated with anomalous PID reports, or PID reports with distance values above a specified value $\varepsilon_{\rm PID}$, and, upon command from the controller, employs the targeting rule to select the process to be killed. For example, the actuator can note the time associated with each anomalous PID report, and then target the process reported most recently, denoted by $\sigma_{\rm PID} = {\rm RCNT}$. Alternatively, the actuator can keep a running average of all anomalous distances reported per process, and then target the process with highest average distance, denoted by $\sigma_{\rm PID} = {\rm DIST}$.

The recovery actuator, RCVR, indicates to the security administrator that manual intervention is desired, and halts the host, denying further propagation of the worm to other vulnerable hosts. Within the scope of our experimentation, this actuator simply posts a message to the terminal. We assume that prematurely selecting RCVR is viewed as no less detrimental to survivability objectives than permitting the final phase of the attack to ensue.

C. Data Collection

While operating in experimentation mode, the prototype ADS logs three key types of events to a file in support of offline model/strategy development for the controller as well as subsequent online validation.

- *Sensor events* correspond directly to the measurement reports sent to the controller by each of the three available CylantSecure streams: KER, PID & IP.
- *Response events* correspond directly to the commands sent by the controller to each of the three available actuators: the passive DFLT parameterized by per-stream staging schemes (τ_ℓ, ρ_ℓ) and detection schemes γ_ℓ^*, the active KILL further parameterized by targeting scheme $\sigma_{\rm PID}$ & anomalous-process indicator $\varepsilon_{\rm PID}$, and the passive RCVR.
- *Truth events* correspond to special markers which reveal the true state trajectory (i.e., the true activity of the remote client machines and the attacker), and are inserted strictly for offline model development & experimental evaluation. Messages marking truth events are sent to the controller by the load generation utilities on the client machines, from which the actual loading condition placed on the web server can be derived, and by the specially instrumented attack script to indicate the beginning and ending of each attack step. We denote these two categories of truth events by NRML and ATCK, respectively. While the controller can reliably mark the initiation of any response, note that truth events reliably marking the actual outcome of an active response (e.g., whether a KILL response has thwarted the attack) can require further instrumentation of the attack script, or perhaps special messages from the actuators themselves.

Table II lists the parameterizations assumed for all of these key event types during our experimentation. During actual operation, sensor & response events are directly observable by the decision-making components of the feedback controller; whereas truth events, which would obviously not be present except during our experimentation, are assumed completely unobservable. Truth events simply allow all sensor events to be associated with a particular state/control pair of a PO-MDP model, which is essential to extract the probability parameters from a collected data set during offline controller design. Furthermore, for any single data run during online controller experimentation, knowledge of the true state trajectory is also essential for computing the associated evaluation metrics defined by (4) & (5) in Section II.

In design mode, the purpose of data collection is to yield a *training set* from which empirical development of a PO-MDP model, including binary threshold selections, can proceed. The particular control strategy is not relevant during training set generation; rather, it is controlled only in the sense of producing an adequate number of samples under all loading conditions & instrumented attack steps, as well as active actuation. For each loading condition, a training set includes at least one long run of normal operation (on the order of twenty minutes), and many repeated runs of attack operation (each on the order of a couple minutes, with the first two phases occurring within tens of seconds). In each attack run, the attack starts randomly relative to the underlying normal operation. In experimentation mode, once an operational model and strategy are available, the purpose of data collection is to support controller evaluation.

$$J(\gamma_\ell) = \sum_{i=1}^{n} \Pr(x_k = i)\left[C(0,i)\Pr(D \le \gamma_\ell | x_k = i) + C(1,i)\Pr(D > \gamma_\ell | x_k = i)\right] \tag{9}$$

TABLE II
KEY EVENTS AND PARAMETERIZATIONS FOR DATA COLLECTION IN EXPERIMENTATION

Event Tag	Data Parameters					
			Sensor Events			
KER	time stamp	distance				
PID	time stamp	distance	process ID	process name	user name	user ID
IP	time stamp	distance	source IP	source port	destination IP	destination port
			Response Events			
DFLT	time stamp	staging scheme	detection scheme			
KILL	time stamp	staging scheme	detection scheme	targeting scheme	target process	
RCVR	time stamp	staging scheme	detection scheme			
			Truth Events			
NRML	time stamp	client ID	load setting			
ATCK	time stamp	step ID	step name			

IV. EXPERIMENTATION RESULTS

To operate the prototype ADS within the described experimentation scenario, we must yet specialize the stationary PO-MDP formulation in Section II to support the two main decision-making components of the feedback controller:

1) the recursive estimator driven by the statistical system & observation models, $\mathbf{F}$ & $\mathbf{H}$, respectively, and
2) the response selector driven by its policy, μ.

The training set from which the PO-MDP model is empirically developed concerns only the "low" workload condition on the web-server and, during attack operation, the worm variant that utilizes the **nmap** scan & **wu-ftp** exploit. The resulting controller is shown to achieve sub-second response, successfully diagnosing the attack's key stages, and, most importantly, preventing the host from being transformed into an attacker that would further propagate the worm. This result is accomplished even for a variant of the worm not appearing in the training set, namely one utilizing the **named** exploit. *We have demonstrated that the prototype ADS defeats an attack that it has never previously seen*. We also illustrate the superiority, from a survivability standpoint, of the feedback control approach over an ad-hoc rule-based approach which ignores uncertainty in the sensor observations & response outcomes.

A. Model/Strategy Development

The first key step is to establish the structure of the PO-MDP model by specifying the state space S of cardinality n, the control space C of cardinality m, and the observation space Z of cardinality q. The second key step is to specify the mn^2 state transition probabilities $\Pr(x_{k+1}|x_k, u_k)$, and the mnq sensor observation probabilities $\Pr(z_k|x_k, u_{k-1})$ via a combination of engineering assumptions & empirical approximation from a training set of collected data. The third step, assuming the statistical models are at least partially controllable & observable, is to populate the mn expected cost-to-go values composing response policy μ via numerical optimization, given $\mathbf{F}$ and $\mathbf{H}$, as well as the mn^2 state transition costs $c(x_k, u_k, x_{k+1})$ stored in a cost function $\mathbf{G}$.

1) Structural Specification: Insisting that a choice for S directly reflect the true, low-level state of the information system being protected is neither sensible for computer security objectives nor practical for a feedback control approach. Instead, S must abstract the security environment, both the modes of anticipated normal activity & the steps of anticipated attack activity, at a level of detail consistent with the capabilities of the available security assets. Denote by L, M, & H the "low," "medium," & "high" server workload conditions, respectively, as the normal modes of operation characterized in Section III-A. The attack steps can start as fine-grained as the ATCK event markers injected into the collected data by the instrumented attack script. However, we appeal to Table I; and denote

- the first two phases of the attack by states $A_1, A_2, \cdots, A_4$,
- the final phase of the attack by F, and, in addition,
- the absence of all attack activity by state N.

More specifically, assuming a constant workload:

- A transition from N to A_1 corresponds to the beginning of the "scan" step, initiating the first phase of the automated attack.
- A transition from A_1 to A_2 corresponds to the beginning of the access control "bypass" step.
- Simultaneously ending the first phase of the attack and beginning the second phase, a transition from A_2 to A_3 corresponds to the first command executed from the successfully attained user shell, typically initiating the "download" of the payload to the victim.
- A transition from A_3 to A_4 corresponds to the "installation" of the payload on the victim.

- Finally, a transition from A_4 to F corresponds to the security failure, entering the final phase of the attack at which time the victim begins to attack other vulnerable machines in order to propagate the worm.

In summary, recognizing that an attack can occur regardless of the current server workload, we choose a state space $S = \{L, M, H\} \times \{N, A_1, ..., A_4, F\}$, or $n = 18$.

We anticipate that different actuator parameterizations are generally going to result in different per-stage transition/observation probabilities and maintenance costs. Let W_N and W_A be two passive controls, with nominal and alternative parameterizations respectively, for the DFLT actuator. Both utilize the same staging scheme for the three measurement streams, namely an update period of 0.2 sec and an aggregation rule which averages all raw distances received within a single period; that is, we set

$$(\tau_{\mathrm{KER}}, \rho_{\mathrm{KER}}) = (\tau_{\mathrm{PID}}, \rho_{\mathrm{PID}}) = (\tau_{\mathrm{IP}}, \rho_{\mathrm{IP}}) = (0.2, \mathrm{AVG}).$$

Both of these controls employ the same detector cost function which expresses a minimum probability-of-error criterion where errors correspond to outputting "alerts" while in state N, or "safes" when not in state N:

$$C(Y_0, i) = \begin{cases} 5, & Y_0 = 1, i = N \\ 1, & Y_0 = 0, i \in \{A_1, ..., A_4, F\} \\ 0, & \text{otherwise} \end{cases} . \qquad (10)$$

However, control W_N optimizes the thresholds assuming an equally-likely prior probability distribution while W_A biases the priors associated with the intermediate attack states to be proportional to the number of measurement samples in the training set associated with that state. In effect, this bias will capture the relative duration of the intermediate states of the worm variant used to generate the training set.

Let control K be tied to the active KILL actuator, utilizing the same staging and detection schemes as for W_A and a targeting scheme against the most recent anomalous process reported on the PID stream, or $(\sigma_{\mathrm{PID}}, \varepsilon_{\mathrm{PID}}) = (\mathrm{RCNT}, 10^{-10})$. Control R also shares the same staging and detection scheme as W_A but represents the initiation of the RCVR actuator, which within the scope of our initial experimentation acts as the purely passive indicator of a detected failure. In summary, we choose a control space $C = \{W_N, W_A, K, R\}$, or $m = 4$.

Choosing a finite observation space Z should reflect the three CylantSecure measurement streams which each output a real-valued "distance" per measurement, KER, PID and IP, coupled with the detection schemes employed for all controls. Assuming a binary detection scheme applied to each stream independently, the KER stream can be either below or above its threshold, denoted by KER:0 or KER:1, respectively. Similarly, the thresholded PID stream produces two possible values, PID:0 and PID:1, and the thresholded IP stream produces two possible values, IP:0 and IP:1. Thus, $Z = \{\mathrm{KER}:0, \mathrm{KER}:1, \mathrm{PID}:0, \mathrm{PID}:1, \mathrm{IP}:0, \mathrm{IP}:1\}$, or $q = 6$.

2) Statistical Parameterization: Given the structural specification of a PO-MDP, the second step is to determine numerical values for the per-stream, control-dependent measurement distributions $\Pr(D|x_k, u_{k-1})$ (so that control-dependent sensor thresholds can be optimized) as well as the state transition probabilities $\Pr(x_{k+1}|x_k, u_k)$ and sensor observation probabilities $\Pr(z_k|x_k, u_{k-1})$. Theoretically, these conditional probabilities are interpreted as the relative frequency, over an infinite number of trials, of each random event given each state/control pair defined by the state & control spaces S & C. Practically, many of the transition probabilities, and all of the measurement distributions & observation probabilities can be empirically approximated from the training set. Clearly, for each state/control pair, the more samples present in the training set, the more likely the empirical approximation satisfies the theoretical interpretation. Given a training set, empirically approximating the probability distributions is as straightforward as counting the number of occurrences of all events represented by each conditional distribution.

Recall that the training set concerns only the "low" workload condition on the web-server. Thus, during normal operation, state (L, N) is the only state encountered. During attack operation, the five states $L \times \{A_1, \ldots, A_4, F\}$ are all sampled and, under passive controls W_N, W_A & R, the only potentially nonzero transition probabilities are for self-transitions in all states, as well as transitions between successive steps of the automated worm attack, remaining in state (L, F) indefinitely upon completion of the attack. Under active control K, the other potentially nonzero transition probabilities are for transitions from $L \times \{A_1, A_2, A_3, A_4, F\}$ back to state (L, N). Fig. 4 shows the state transition probabilities empirically obtained from the available training set, concerning only the attack variant which utilizes the **nmap** scan & **wu-ftp** exploit. The nonzero probabilities of transitioning from states $L \times \{A_2, A_3, A_4\}$ back to state (L, N) for active control K imply at least a partially controllable model $\mathbf{F}$. Fig. 5 shows the empirical results of applying (8)–(10) to derive the binary thresholds per measurement stream using the available training set. Because the distributions differ noticeably across the states $L \times \{A_1, A_2, A_3, A_4, F\}$, we can conclude to have at least a partially observable model $\mathbf{H}$.

The only statistics which cannot be *reliably* approximated from training data correspond to the transition from the absence of an attack, or state (L, N), to initiation of the attack. In other words, we must specify the conditional distribution $\Pr(x_{k+1}|x_k = (L, N), u_k)$ using engineering assumptions which reflect a "best guess" as to the likelihood of experiencing an attack. Letting parameter p_A represent the per-stage likelihood of being attacked, and assuming a fixed "low" workload as well as the class of automated worm-like attacks, we select

$$\Pr(x_{k+1}|x_k = (L, N), u_k) = \begin{cases} 1 - p_A, & x_{k+1} = (L, N) \\ p_A, & x_{k+1} = (L, A_1) \\ 0, & \text{otherwise} \end{cases} . \qquad (11)$$

During operation, the numerical value assigned to probability p_A directly impacts the readiness of the estimator to infer the presence of attack activity based on sensor alerts. Overall, the

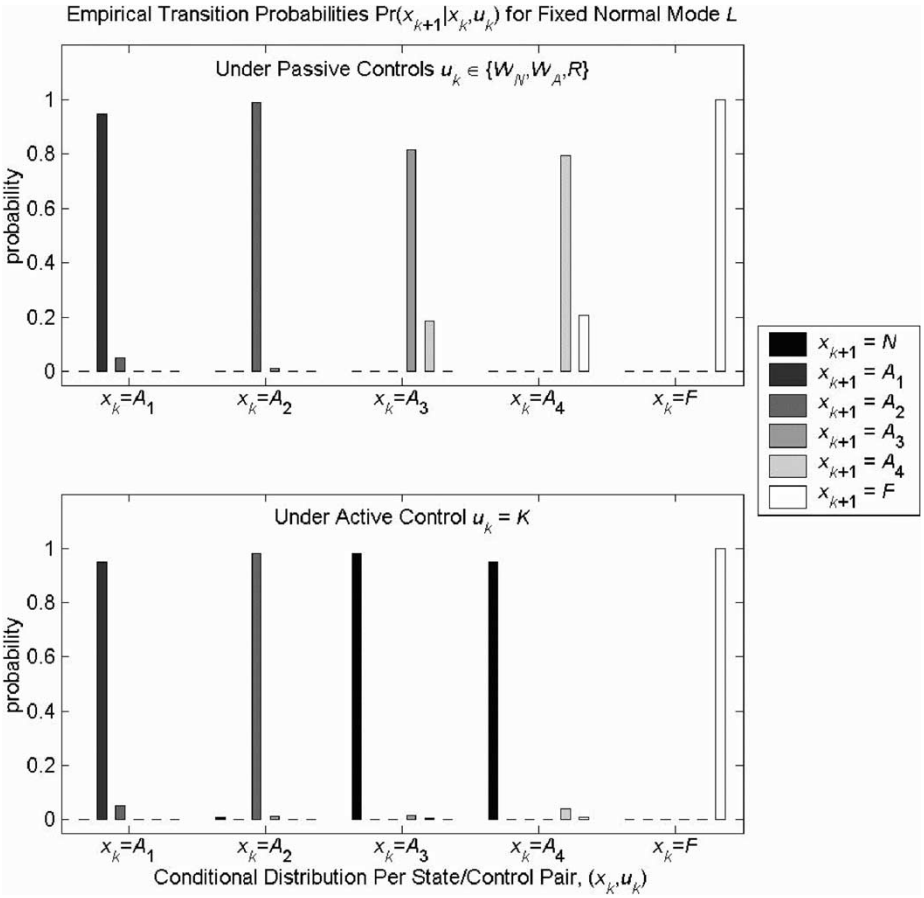

Fig. 4. System model used during experimentation: normal workload fixed at "low" level; attack using **nmap** scan & **wu-ftp** exploit.

better the sensors can observe the attack activity, the less dependent controller performance becomes on the assigned numerical value for the typically unknown parameter p_A.

3) Policy Selection: Ideally, given system model $\mathbf{F}$ & observation model $\mathbf{H}$, expressing the cost function $\mathbf{G}$ allows a dynamic programming algorithm to automatically generate the optimal response policy. Beyond the interpretation of cost to the administrator, the absolute values are not critical because the form of the response policy recognizes cost tradeoffs based only on the relative values. Practically, however, the complexity associated with its exact solution is prohibitive; and approximation methods must be engineered, which we consider beyond the scope of our initial experimentation. Instead, we employ a heuristic response policy whose numerical parameters are manually selected prior to controller operation. In this case, the cost function $\mathbf{G}$ is only relevant for computing the primary evaluation metric in (5) for any data run during experimentation.

Our response policy assumes the appropriate control is W_N when the process is truly in state N. At the other extreme, when in state F, the policy assumes the appropriate control is R. During the intermediate attack states, let W_A be the appropriate control when truly in state A_1; and K be the appropriate control when in any one of the partially-controllable states, A_2, A_3, or A_4. Recall that the execution of active control K depends on the KILL actuator being aware of at least one anomalous process which can be targeted; if no such process is available, the control defaults to W_A. A cost function G which reflects these assumptions is populated by transition costs

$$c(x_k, u_k, x_{k+1}) = \begin{cases} c_1, & x_k \in \{A_1, A_2\} \text{ and } u_k = W_N \\ c_2, & x_k \in \{A_3, A_4\} \text{ and } u_k \in \{W_N, W_A\} \\ c_3, & x_k = F \text{ and } u_k \neq R \text{ or } x_k \neq F \text{ and } u_k = R \\ c_X, & x_k \in \{N, A_1\} \text{ and } u_k = K \\ c_K, & x_k \in \{A_2, A_3, A_4\} \text{ and } u_k = K \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

where

i) parameters c_1 & c_2 reflect the "failure" cost associated with allowing a single stage of the first & second phases, respectively, of the attack to proceed without selecting appropriate control,

ii) parameter c_3 reflects the "failure" cost & equivalent "maintenance" cost associated with inappropriate nonselection & selection, respectively, of passive control R, and

iii) parameters c_X & c_K reflect the "maintenance" cost associated with the inappropriate & appropriate selection, respectively, of active control K.

We consider the intuitive class of policies μ which select out of the $m = 4$ candidate controls based on three independently selected *sensitivity parameters*, $\eta_A, \eta_K, \eta_R \in (0, 1)$, carefully

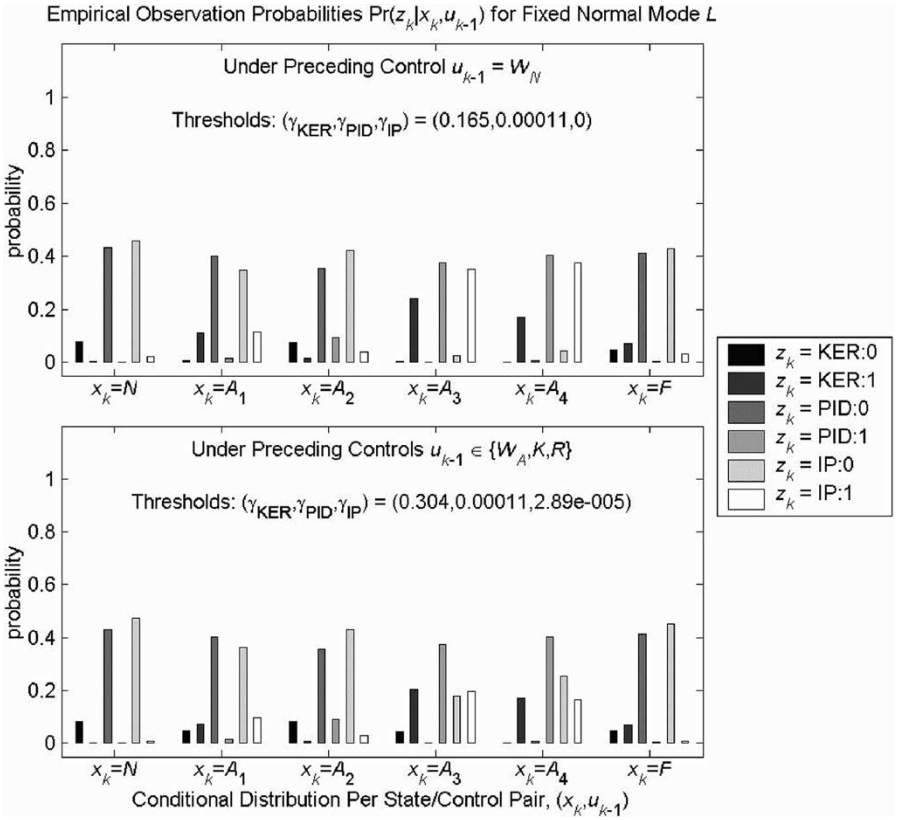

Fig. 5. Observation model used during experimentation: normal workload fixed at "low" level; attack using **nmap** scan & **wu-ftp** exploit.

applied to the components of the probabilistic state estimate. More specifically, apply control W_N as long as the estimator's s-confidence in state N exceeds probability $1-\eta_A$; otherwise, considering the s-confidence in state A_1 relative to the total s-confidence in all other attack states, apply control W_A provided this relative s-confidence exceeds probability $1-\eta_K$; otherwise, considering the total s-confidence in controllable states A_2, A_3 or A_4 relative to the s-confidence in state F, apply control K provided this relative s-confidence exceeds probability $1-\eta_R$; otherwise, apply control R. Mathematically, as a function of the $n = 6$ components of the probabilistic state B_k and the three sensitivity parameters (η_A, η_K, η_R), this heuristic policy is expressed by

$$\mu(B_k) = \begin{cases} W_N, & \eta_A \le [B_k]_1 \le 1 \\ W_A, & [B_k]_1 < \eta_A \text{ and } \eta_K \le \frac{[B_k]_2}{(1-[B_k]_1)} \le 1 \\ K, & [B_k]_2(1-[B_k]_1) < \eta_K \text{ and } \eta_R \le \frac{b_K}{[B_k]_6} \\ R, & \text{otherwise} \end{cases} \tag{13}$$

where $b_K = [B_k]_3 + [B_k]_4 + [B_k]_5$ represents the total estimator s-confidence in any one of the partially-controllable states. Note that a lower numerical value for a sensitivity parameter implies the estimator's s-confidence must exceed a higher probability threshold before the corresponding control is triggered. In short, a lower sensitivity implies a less likely control.

B. Controller Evaluation

Given the statistical models depicted in Figs. 4 and 5, and the policy of the form expressed in (13), we are ready to operate the prototype ADS in the laboratory web-server environment. In all data runs which follow, the normal workload generated on the server is fixed at the "low" configuration, matching the training set from which the PO-MDP model was derived. However, as highlighted above, we will experiment with a worm variant which is distinct from any attack contained in the training set. We assign a numerical value of $p_A = 10^{-6}$ to the unknown probability of being attacked, and initialize the inputs to the recursive estimator to be $u_{-1} = W_N$ and $B_{-1} = [1-p_A \quad p_A \quad 0 \quad 0 \quad 0 \quad 0]'$. Finally, so that the average cost per stage can be computed for each run, the following numerical values are assigned to the cost parameters listed in (12): $c_1 = 10$, $c_2 = 100$, $c_3 = 1000$, $c_X = 50$ and $c_K = 20$.

Fig. 6 shows a total of eighteen separate time-tagged data series related to a single ADS experiment, where the origin on the time axis is purposely marked sufficiently after the fading of any start-up transients in emulating normal web traffic & user sessions on the laboratory network. The vertical scale of each time series, though not explicitly labeled, is from zero to one. Most of these time series represent the inputs & outputs of the controller process, but a few also represent unobservable information to support experimental evaluation.

- The top three time series, labeled **Streams**, are the raw, nonnegative distance values reported by each of CylantSecure's streams. Most of these distances fall below unity,

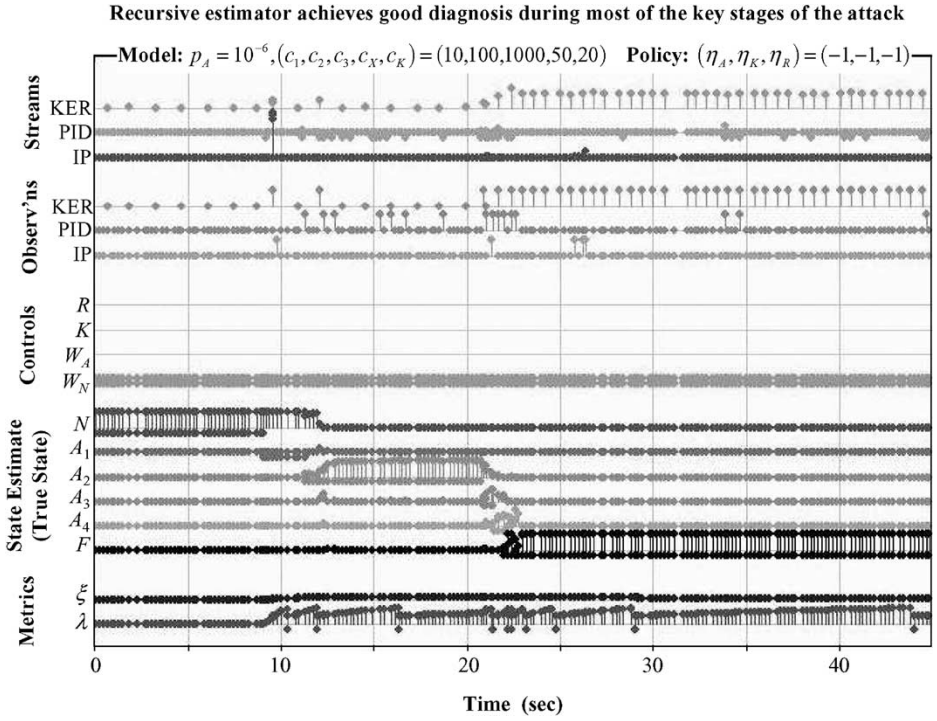

Fig. 6. ADS experimentation: normal workload at "low" level; attack using **nmap** scan and **wu-ftp** exploit; feedback controller using "uncontrolled" policy.

but occasionally a distance report can spill into a neighboring time series (e.g., the IP distance approximately nine seconds into the data run). The occasional negative-valued stem along the PID stream indicates that an anomalous process, whose distance exceeded the value of ε_{PID}, was added to the list of potential target processes maintained by the KILL actuator.

- The next three series, labeled **Observations**, are the inputs to the recursive estimator, and correspond to the finite-valued sequence generated from the real-valued measurement sequence via the staging & detection schemes.
- The next four time series, labeled **Controls**, is the output of the response selector which, under the "uncontrolled" policy, is fixed at W_N.
- Finally, the next six time series, labeled **State Estimate**, are the nonnegative-valued components of the probabilistic state vector in each decision stage. Though not observable by either decision-making component of the controller, the negative-valued stems in the estimate output indicate the true state trajectory. For example, in Fig. 6, the process begins in state N, and the attack initiates after approximately nine seconds of normal operation, indicated by the negative-valued stems in the time series for the first attack state A_1. Attack state A_2 begins approximately two seconds later, state A_3 approximately ten seconds later, and so on.
- The final two time series, labeled **Metrics**, are the empirical evaluation of the average error per stage ξ and average cost per stage λ defined in (4) & (5), respectively. In Fig. 6, we see that the estimator achieves good diagnosis on most of the key stages of the attack, and therefore ξ remains relatively low. However, because of the absence of control, the attack proceeds to completion, and large "failure" costs accrue. For the assigned numerical cost values, note that λ can easily exceed a value of unity. The negative-valued stems on the time series for λ represent a doubling of the vertical scaling; thus, the saw-tooth nature of this time series in Fig. 6, coupled with negative-valued stems, represents monotonically increasing cost.

1) ADS Thwarts Attack Objectives in Real Time: Fig. 7(a) depicts controller operation under the "heuristic" policy, parameterized by sensitivity values $(\eta_A, \eta_K, \eta_R) = (0.5, 0.1, 0.1)$, for the exact same sensor input as shown in Fig. 6. In other words, the controller is operating on previously logged data originally generated under an "uncontrolled" policy, and, thus, actuation of any selected active control can have no effect. While Fig. 7(a) indicates where active control K would have been selected under the "heuristic" policy, the true response as far as future decision-making is concerned is K's passive counterpart, control W_A. The negative-valued stems on the time series labeled **Controls** convey precisely this notion; that is, when considering historical data with a control policy which differs from the policy used during the initial generation of the data, a distinction between selected responses and actuated responses exists for active controls. The negative-valued stems correspond to the actuated responses recorded during live operation. Note that the distinction does not arise with purely passive controls.

A comparison between Figs. 6 and 7(a) illustrates the potential that the ADS provides for improving host survivability. Not only would ADS initiate kills in the appropriate steps of the attack, but it also quickly diagnoses the costly state F as indicated by the selection of control R only a few seconds upon compromise. This could prevent the compromised host

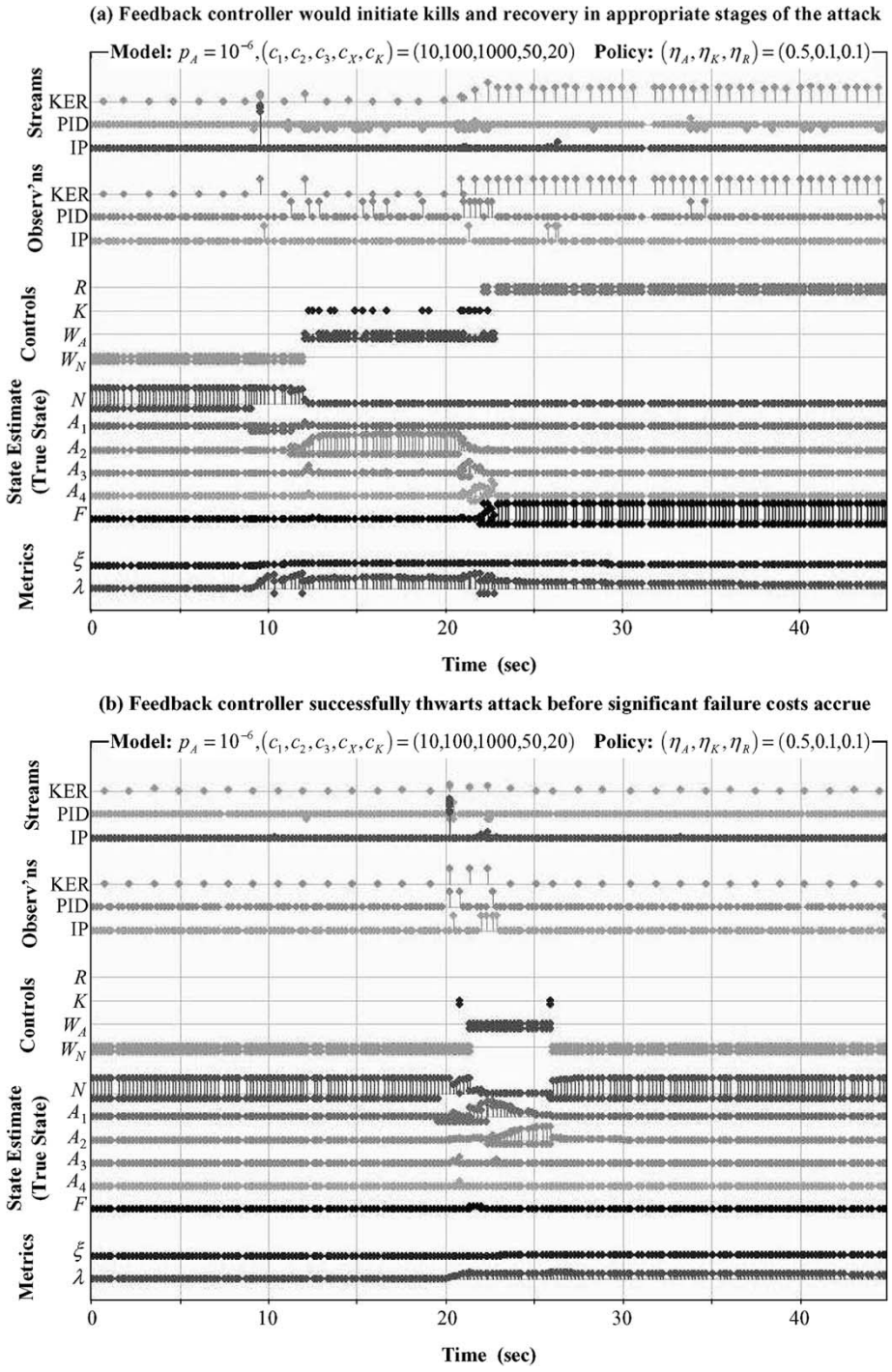

Fig. 7. ADS experimentation: normal workload at "low" level; attack using **nmap** scan & **wu-ftp** exploit; controller using "heuristic" policy with actuation (a) disabled and (b) enabled.

from further propagating the worm to other vulnerable hosts. Based on the cost function, these benefits are quantified by the reduced average cost per stage, or metric λ, in Fig. 7(a), compared to Fig. 6.

Fig. 7(b) shows the output of a live data run where actuation is enabled, using the same policy as used in Fig. 7(a). The ADS successfully thwarts the attack before it completes the bypass, and gains a privileged shell. In this particular data run, note that it also prematurely applied control K, and mistakenly targeted a friendly process while that attack was still in the scan state, A_1. It was the unusual density of sensor alerts while in state A_1 that misled the controller, but it subsequently thwarts the attack, and correctly recognizes the return of normal operation. Note again that, compared to Figs. 6 and 7(a) and (b) shows a significantly lower evaluation of the cost metric.

2) ADS Defeats Previously Unseen Attacks : Fig. 8(a) and (b) are the analogous illustrations to Fig. 7(a) and (b), except that the attack uses the **named** exploit, a variant of the worm not present in the training set. A comparison of Figs. 7(a) and 8(a) give a sense of how the two attack variants differ. The worm

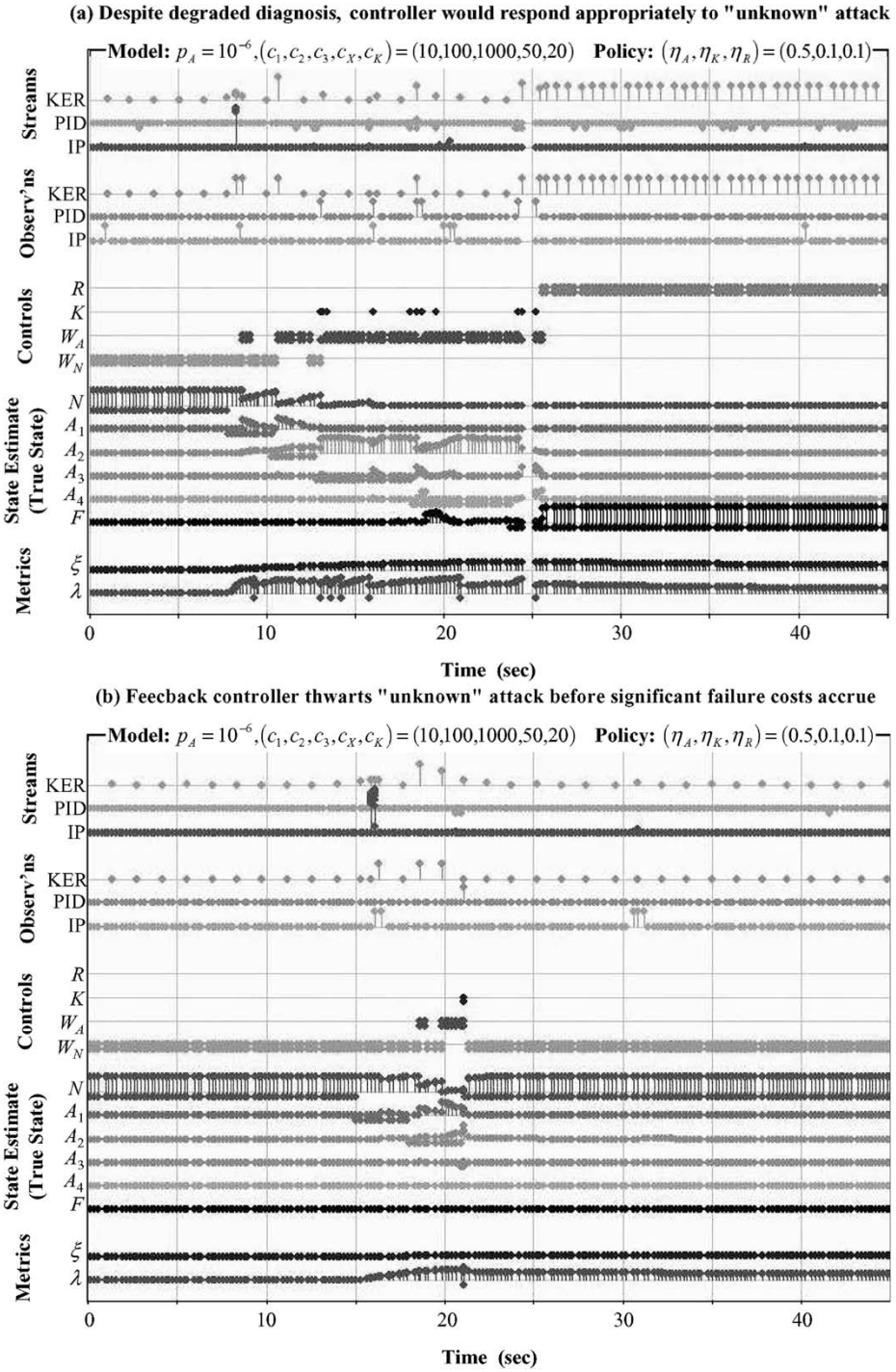

Fig. 8. ADS experimentation: normal workload at "low" level; attack using **nmap** scan & **named** exploit; controller using "heuristic" policy with actuation (a) disabled and (b) enabled.

using the **named** exploit exhibits a shorter bypass step, or time spent in state A_2, but longer download and install steps as conveyed by the time spent in states A_3 and A_4, respectively. In addition, the density of anomalous process activity as reported by the PID stream seems to be lower for the **named** exploit than for the **wu-ftp** exploit. Though Fig. 8(a) shows degraded attack step diagnosis & a corresponding increase in the error metric ξ, the controller still initiates kills in the appropriate states & quickly indicates the state of failure with control R. Fig. 8(b) shows the output for a live data run of the **named** attack variant, with the controller killing the fraudulently obtained shell privilege immediately upon completion of the bypass.

3) ADS Overcomes Limitations of Ad-Hod Response: Figs. 7 and 8 have illustrated how an ADS driven by a feedback controller can reduce the "failure" cost incurred by thwarting attack objectives in real-time. Another benefit of feedback control is how it addresses the limitations identified with ad-hoc response schemes. As discussed in Section I, automated response based on static rules which typically consider single alerts at a time, especially when considering anomaly detectors, have been

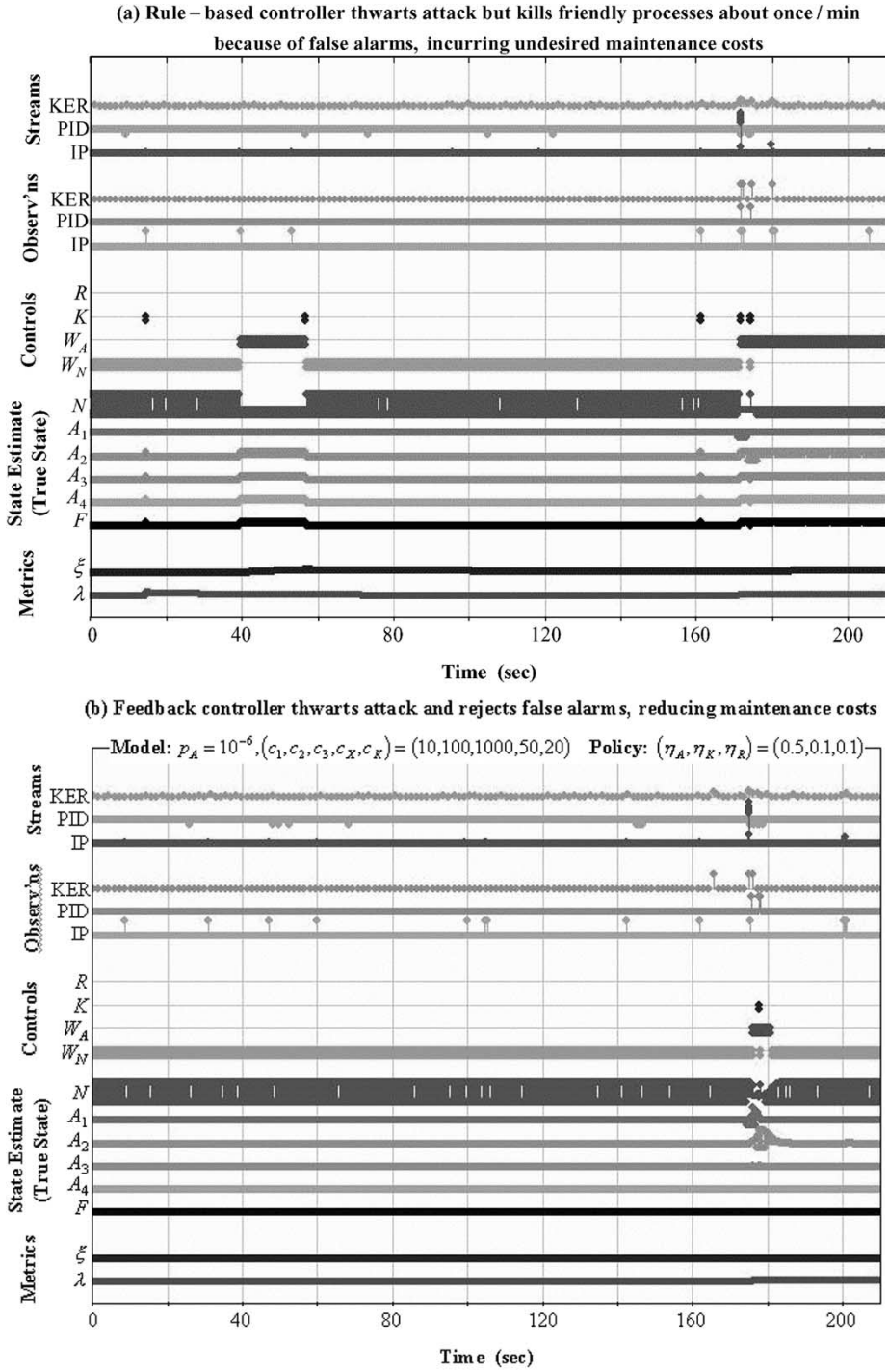

Fig. 9. ADS experimentation: normal workload fixed at "low" level; attack using **nmap** scan & **wu-ftp** exploit; comparison of (a) rule-based controller & (b) feedback controller.

plagued by intolerable rates of false alarms, leading to costly over-application of security-related defenses.

We assume a rule-based controller that is equipped with exactly the same sensor & actuators as the feedback controller. However, it does not use the PO-MDP model to make decisions, but rather ignores any uncertainty associated with the sensor alerts as well as the outcome of the active controls. It uses the following simple rules:

i) every sensor alert toggles the confidence from being in state N to being under attack,
ii) while confident of being under attack, the application of active control K toggles the confidence back to being in state N, and
iii) while confident of being under attack, if the PID stream has not yet identified an anomalous process to be targeted, passive control W_A is applied until an anomalous process is identified.

Fig. 9(a) and (b) show the output of the rule-based controller & feedback controller, respectively, during an extended period of normal operation after which point an attack is initiated.

While both controllers successfully thwart the attack, during the three minutes of normal operation which preceded the attack, the rule-based controller targets a friendly process on three separate occasions, whereas the feedback controller successfully rejects all of the false alarms. The feedback controller reduces the rate of inappropriately applied response, resulting in a reduced "maintenance" cost.

V. Conclusion

While the offline design effort required in support of feedback control might exceed that required to support rule-based control, a feedback control approach includes a systematic, empirically-driven design methodology which ultimately realizes significant gains in online effectiveness, & consequently, overall information system survivability. The experimentation results presented here are only single sample paths of a modeled statistical process; thus, while the feasibility and potential of our approach is certainly established, a more comprehensive assessment demands experiments which measure the survivability gain with statistical significance. Future plans also include further experimentation to investigate the impact of client loading on the web server, the impact of more intelligent or persistent worm-like attacks, and the development of policy generation algorithms which apply to more elaborate sets of security assets against multiple types of attacks. Longer-term questions concern the feasibility of a feedback control approach, demonstrated in this work only for host-based security, to network-level & router-level information security.

Acknowledgment

Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Space and Naval Warfare Systems Center-San Diego.

The authors wish to thank Dr. M. Bajura, S. Carter, Dr. G. Frazier, J. O'Neill and C. Koch for their tireless efforts with the laboratory setup, implementation and experiments. Dr. Bajura also contributed to several of the opinions and findings expressed in the body of this paper and Dr. Frazier also contributed to editing the final manuscript. The authors also thank S. Wimer and M. Wimer for their guidance regarding calibration & operation of the CylantSecure system utilized in the laboratory experimentation. Dr. C. Lawrence, Dr. K. Mullin, Prof. V. Gligor, Prof. S. Marcus, Prof. J. Munson, Dr. C. Morefield, Dr. C. McCollum, and Dr. H. Jones all provided influential discussions on this subject.

References

[1] (2002) CERT/CC Statistics 1988–2001. CERT Coordination Center, PA. [Online]. Available: http://www.cert.org/stats/cert_stats.html

[2] J. Howard, "An Analysis of Security Incidents on the Internet 1989–1995," Ph.D., Carnegie Mellon University, Engineering and Public Policy, 1997.

[3] J. H. Houle and G. M. Weaver, "Trends in Denial of Service Attack Technology," Carnegie Mellon University, CERT Coordination Center, 2001.

[4] M. Deutsch and R. Willis, *Software Quality Engineering: A Total Technical and Management Approach*. Englewood Cliffs, NJ: Prentice-Hall, 1988.

[5] M. Barbacci, M. Klein, T. Longstaff, and C. Weinstock, "Quality Attributes," Carnegie Mellon University, Software Engineering Institute, Pittsburgh, PA, CMU/SEI-95-TR-02, 1995.

[6] A. Avizienis, J. Laprie, and B. Randell, "Fundamental concepts of dependability," in *Proc. of the Information Systems Survivability Workshop*, Boston, MA, 2000, pp. 7–12.

[7] J. Lauber, C. Steger, and R. Weiss, "Autonomous agents for online diagnosis of a safety-critical system based on probabilistic causal reasoning," in *Proc. of the 4th Int. Sym. on Autonomous Decentralized Systems*, Tokyo, Japan, 1999.

[8] A. Pouliezos and G. Stavrakakis, *Real-Time Fault Monitoring of Industrial Processes*. Boston, MA: Kluwar Academic Publishers, 1994.

[9] B. Schroeder, K. Schwan, and S. Aggarwal, "Software approach to hazard detection using online analysis of safety constraints," in *Proc. of the 16th Int. Sym. on Reliable Distributed Systems*, Durham, NC, 1997, pp. 80–87.

[10] A. Avizienis, *Toward Systematic Design of Fault-Tolerant Systems*: IEEE Computer, 1997, pp. 51–58.

[11] S. Axelsson, "Intrusion Detection Systems: A Survey and Taxonomy," Chalmers University of Technology, Dept. of Computer Engineering, Sweden, 99-15, 2000.

[12] J. Doyle, I. Kohane, W. Long, H. Shrobe, and P. Szolovits, "Event recognition beyond signature and anomaly," in *Proc. of the IEEE Workshop on Information Assurance and Security*, West Point, New York, 2001, pp. 17–23.

[13] U. Lindqvist and P. A. Porras, "eXpert-BSM: A host-based intrusion detection solution for sun solaris," in *Proc. 17th Annual Comp. Security Applications Conf.*, 2001, pp. 240–251.

[14] A. Warrender, S. Forrest, and B. Pearlmutter, "Detecting intrusions using system calls: Alternative data models," in *Proc. IEEE Sym. on Security and Privacy*, 1999, pp. 133–145.

[15] S. Elbaum and J. Munson, "Intrusion detection through dynamic software measurement," in *Proc. of the Workshop on Intrusion Detection and Network Monitoring*, Santa Clara, CA, 1999, pp. 41–50.

[16] J. Allen, A. Christie, W. Fithen, J. McHugh, J. Pickel, and E. Stoner, "State of the Practice of Intrusion Detection Technologies," Carnegie Mellon University, Software Engineering Institute, Pittsburgh, PA, CMU/SEI-99-TR-028, 1999.

[17] R. P. Lippmann *et al.*, "Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation," in *Proc. of the DARPA Information Survivability Conference and Exposition: DISCEX-2000*, vol. 2, Hilton Head Island, SC, January 2000.

[18] B. Bertsekas, *Dynamic Programming and Optimal Control*. Belmont, MA: Athena Scientific, 1996, vol. I.

[19] M. Athans and P. Falb, *Optimal Control*. New York, NY: McGraw-Hill, 1966.

[20] H. Van Trees, *Detection, Estimation and Modulation Theory Part I*. New York, NY: John Wiley and Sons, 1968.

[21] J. Egan, *Signal Detection Theory and ROC Analysis*, NY: Academic Press, 1975.

[22] J. Tsitsiklis, "Decentralized detection," *Advances in Statistical Signal Processing*, vol. 2, pp. 297–344, 1993.

[23] A. Gelb, J. Kasper, R. Nash, C. Price, and A. Sutherland, *Applied Optimal Estimation*. Cambridge, MA: MIT Press, 1974.

[24] K. Shanmugan and A. Breipohl, *Random Signals: Detection, Estimation and Data Analysis*. New York, NY: John Wiley & Sons, 1988.

[25] M. Putterman, *Markov Decision Processes*. New York, NY: John Wiley & Sons, 1994.

[26] W. Lovejoy, "A survey of algorithmic methods for partially observable Markov decision processes," *Annals of Operations Research*, vol. 28, pp. 47–66, 1991.

[27] C. Fernandez-Gaucherand, A. Arapostathis, and S. Marcus, "On the average cost optimality equation and the structure of optimal policies for partially observable Markov decision processes," *Annals of Operations Research*, vol. 29, pp. 471–512, 1991.

[28] B. Papadimitiou and J. Tsitsiklis, "The complexity of Markov decision processes," *Math. Operations Research*, vol. 12, pp. 441–450, 1987.

[29] R. Gallager, *Discrete Stochastic Processes*. Boston, MA: Kluwar Academic, 1996.

[30] C. Fernandez-Gaucherand and S. Marcus, "Risk-sensitive optimal control of hidden Markov models: Structural results," *IEEE Trans. on Automatic Control*, vol. 42, pp. 1418–1422, 1997.

[31] R. Durst, T. Champion, B. Witten, E. Miller, and L. Spagnuolo, "Testing and evaluating computer intrusion detection systems," *Comm. of the ACM*, vol. 42, no. 7, pp. 53–61, 1999.

[32] R. Maxion and K. Tan, "Benchmarking anomaly-based detection systems," in *Proc. of the Inter. Conf. on Dependable Systems and Networks*, New York, NY, 2000, pp. 623–630.

[33] S. Axelsson, "The base-rate fallacy and its implications for the difficulty of intrusion detection," in *Proc. of the 6th Conf. on Computer and Communications Security*, Singapore, 1999, pp. 1–7.

[34] N. Ye, S. Emran, X. Li, and Q. Chen, "Statistical process control for computer intrusion detection," in *Proc. of the DARPA Information Survivability Conf. & Expo.: DISCEX 2001*, vol. I, Anaheim, CA, 2001, pp. 3–14.

[35] N. Ye, J. Giordano, and J. Feldman, "CACS–A process control approach to cyber attack detection," *Communications of the ACM*, vol. 44, no. 8, pp. 76–82, 2001.

[36] M. Cramer, J. Cannady, and J. Harrell, "New methods of intrusion detection using control-loop measurement," in *4th Technology for Information Security Conf.*, Houston, TX, 1996.

[37] C. Musliner, M. Pelican, W. Heimerdinger, R. Goldman, D. O'Brien, and D. Apostal, "Automatically synthesizing computer security control systems," in *13th Annual Conference on Computer Security Incident Handling*, Toulouse, France, 2001.

[38] A. Briney, *2001 Information Security Industry Survey*: Information Security Magazine, 2001, pp. 34–47.

[39] CERT Incident Note IN-2001–01, Widespread Compromises via "Ramen" Toolkit [Online]. Available: http://www.cert.org/incident_notes/IN-2001–01.html

[40] Software Systems International [Online]. Available: http://www.softsysint.com

O. Patrick Kreidl received a B.S. degree in Electrical Engineering (1994, with highest distinction & a Physics minor) from George Mason University (GMU) in Fairfax, Virginia and a M.S. degree in Electrical Engineering & Computer Science (1996) from the Massachusetts Institute of Technology (MIT) in Cambridge, Massachusetts. During the summers of 1996 and 1997, he was a Member of the Technical Staff in the Information Systems Division of Alphatech, Inc., Burlington, Massachusetts. Also during the summer of 1997, he was affiliated with the Institute for Defense Analyses, Alexandria, Virginia, serving as the graduate student member of a DARPA-sponsored Information Sciences & Technology Study Group. From 1998 to 2002, while on leave from MIT's Ph.D. program, he worked as a Senior Analyst for the Arlington Division of Alphatech, Inc., Arlington, Virginia, and also held an adjunct faculty appointment within GMU's Department of Electrical & Computer Engineering. Mr. Kreidl returned to MIT in Fall 2002 and is currently a Research Assistant in the Stochastic Systems Group within the Laboratory for Information & Decision Systems. Mr. Kreidl's research interests include theories and applications of systems, control, estimation, optimization and information. He has prepared papers, reports and presentations addressing a variety of modern engineering problem domains and has received various academic honors, professional awards and service acknowledgments. He is a student member of the IEEE.

Tiffany M. Frazier received a B.S. in Computer Science from the University of Wisconsin at Madison in 1986, and a M.S. & Ph.D. in Computer Science from UCLA in 1991 and 1995. Dr. Frazier is currently the Director of the Advanced Computing group at ALPHATECH, Inc. The Advanced Computing group conducts multi-disciplinary research & development in complex distributed & agent-based systems, including information assurance, machine learning, reliable computing, real-time systems, stochastic control & optimization, as well as computer application & system architecture design & analysis. Prior to joining ALPHATECH in 1999, Dr. Frazier worked on reliable & safety-critical systems for the FAA & DoD, and contributed to advanced systems concepts for real-time tactical routing & control of both military and civilian aircraft. She was previously Chief Technologist of the Software Engineering Center at MITRE, then Chief Engineer of the C3 Systems Engineering, Integration, and Development Group at SAIC. She is a member of ACM and IEEE Computer, Control, and Communications Societies.