NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

DISSERTATION

A MOVING TARGET DEFENSE SCHEME WITH OVERHEAD OPTIMIZATION USING PARTIALLY OBSERVABLE MARKOV DECISION PROCESSES WITH ABSORBING STATES

by

Ashley S. M. McAbee

September 2020

Dissertation Supervisors: John C. McEachen
Murali Tummala

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2020	**3. REPORT TYPE AND DATES COVERED** Dissertation

4. TITLE AND SUBTITLE
A MOVING TARGET DEFENSE SCHEME WITH OVERHEAD OPTIMIZATION USING PARTIALLY OBSERVABLE MARKOV DECISION PROCESSES WITH ABSORBING STATES

5. FUNDING NUMBERS

6. AUTHOR(S) Ashley S. M. McAbee

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT (maximum 200 words)

Moving target defense (MTD) is a promising strategy for gaining advantage over cyber attackers, but these dynamic reconfigurations can impose significant overhead. We propose implementing MTD within an optimization framework so that we seize defensive advantage while minimizing overhead. This dissertation presents an MTD scheme that leverages partially observable Markov decision processes (POMDP) with absorbing states to select the optimal defense based on partial observations of the cyber attack phase. In this way, overhead is minimized as reconfigurations are triggered only when the potential benefit outweighs the cost. We formulate and implement a POMDP within a system with Monte-Carlo planning-based decision making configured to reflect defender-defined priorities for the cost-benefit tradeoff. The proposed system also includes a performance-monitoring scheme for continuous validation of the model, critical given attackers' ever-changing techniques. We present simulation results that confirm the system fulfills the design goals, thwarting 99% of inbound attacks while sustaining system availability at greater than 94% even as probability of attack phase detection dropped to 0.74. A comparable system that triggered MTD techniques pseudorandomly maintained just 43% availability when providing equivalent attack suppression, which illustrates the utility of our proposed scheme.

14. SUBJECT TERMS Markov processes, cyber defense, moving target defense, decision making under uncertainty	**15. NUMBER OF PAGES** 169
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

A MOVING TARGET DEFENSE SCHEME WITH OVERHEAD OPTIMIZATION USING PARTIALLY OBSERVABLE MARKOV DECISION PROCESSES WITH ABSORBING STATES

Ashley S. M. McAbee
Lieutenant Commander, United States Navy
BS, U.S. Naval Academy, 2007
MS, Electrical Engineering, Naval Postgraduate School, 2013

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2020

Approved by:

Murali Tummala
Department of
Electrical and Computer Engineering
Dissertation Supervisor

John C. McEachen
Department of
Electrical and Computer
Engineering
Dissertation Supervisor

David C. Jenn
Department of
Electrical and Computer Engineering

Thor Martinsen
Department of
Applied Mathematics

Preetha Thulasiraman
Department of
Electrical and Computer Engineering

Murali Tummala
Department of
Electrical and Computer
Engineering
Dissertation Chair

Approved by:

Douglas J. Fouts
Chair, Department of Electrical and Computer Engineering

Orrin D. Moses
Vice Provost of Academic Affairs

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Moving target defense (MTD) is a promising strategy for gaining advantage over cyber attackers, but these dynamic reconfigurations can impose significant overhead. We propose implementing MTD within an optimization framework so that we seize defensive advantage while minimizing overhead. This dissertation presents an MTD scheme that leverages partially observable Markov decision processes (POMDP) with absorbing states to select the optimal defense based on partial observations of the cyber attack phase. In this way, overhead is minimized as reconfigurations are triggered only when the potential benefit outweighs the cost. We formulate and implement a POMDP within a system with Monte-Carlo planning-based decision making configured to reflect defender-defined priorities for the cost-benefit tradeoff. The proposed system also includes a performance-monitoring scheme for continuous validation of the model, critical given attackers' ever-changing techniques. We present simulation results that confirm the system fulfills the design goals, thwarting 99% of inbound attacks while sustaining system availability at greater than 94% even as probability of attack phase detection dropped to 0.74. A comparable system that triggered MTD techniques pseudorandomly maintained just 43% availability when providing equivalent attack suppression, which illustrates the utility of our proposed scheme.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

List of Tables

List of Acronyms and Abbreviations

DESPOT determinized sparse partially observable tree planning

HMM hidden Markov model

ICMP internet control message protocol

IDS intrusion detection system

IEEE Institute of Electrical and Electronics Engineers

IP internet protocol

MDP Markov decision process

MTD moving target defense

POMCP partially observable Monte Carlo planning

POMDP partially observable Markov decision process

SARSOP successive approximations of the reachable space under optimal policies

TCP transmission control protocol

VM virtual machine

THIS PAGE INTENTIONALLY LEFT BLANK

List of Symbols

a	action vector, size $1 \times m$
$a(t)$	action selected at time t
$b(t)$	belief state vector at time t, size $1 \times n$
c_{atk}	attack penalty, scalar
c_def	vector of defensive costs, size $1 \times m$
C_i	cost matrix under action a_i, size $n \times n$
$c_{i,j,k}$	cost incurred under action a_i transitioning from state s_j to state s_k, scalar
$c(t)$	cost incurred at time t, scalar
d	simulation tree depth, scalar
$E(t)$	error signal, scalar
f_a	vector of frequencies of occurrence for each element in a, size $1 \times m$
f_ω	vector of frequencies of occurrence for each element in ω, size $1 \times n$
I_x	identity matrix, size $x \times x$
m	number of actions, scalar
n	number of states, scalar
N	fundamental matrix of expected events before absorption, size $(n-1) \times (n-1)$
N_{dg}	diagonal of matrix N, equivalent to $N \circ I$, size $(n-1) \times (n-1)$
N_{sq}	element-wise square of N, equivalent to $N \circ N$, size $(n-1) \times (n-1)$
N_2	matrix of variances in expected state transitions, size $(n-1) \times (n-1)$
O_i	observation probability matrix under action a_i, size $n \times n$
$o_{i,j,k}$	probability of receiving observation ω_k if in state s_j under action a_i, scalar
p_D	probability of detection for the intrusion detection system, scalar
p_f	probability of failure of a defense, scalar
p_{FA}	probability of false alarm for the intrusion detection system, scalar
P_i	state transition probability matrix under action a_i, size $n \times n$
$p_{i,j,k}$	transition probability for moving from state s_j to state s_k under action a_i, scalar
p_M	probability of missed detection for the intrusion detection system, scalar
p_s	probability of success of a defense, scalar
q	particle count, scalar
Q_i	transient state partition of P_i, size $(n-1) \times (n-1)$

q_π	probability decision is optimal, scalar
R	absorption vector, right column partition of P_i, size $(n-1) \times 1$
s	state vector, size $1 \times n$
$s(t)$	state at time t, scalar
t	time step, scalar
u	improvement in state estimate, scalar
U_i	uncertainty profile
V	value of a policy, scalar
z	decision computational time limit, scalar
χ	average cost incurred per decision, scalar
δ	error threshold vector, size 1×3
ϵ	error metric vector, size 1×3
ϵ_a	error metric describing error in f_a, scalar
ϵ_{atk}	expected error metric vector when in state s_n, size 1×3
ϵ_ω	error metric describing error in f_ω, scalar
ϵ_χ	error metric describing error in χ, scalar
η_{atk}	attack volume, scalar
η_{dec}	decision volume in millions, scalar
γ	discount factor balancing immediate and future costs, scalar
λ_i	arrival rate for pseudo-random action a_i, scalar
μ	number of available services, scalar
μ_v	number of vulnerable services, scalar
ν	attack penalty scaling factor, scalar
ω	observation vector, size $1 \times n$
$\omega(t)$	observation at time t, scalar
ϕ	attack suppression, scalar
π	Optimal policy vector for equivalent Markov decision process, size $1 \times n$
Π	optimal policy matrix for equivalent Markov decision process, size $m \times n$
ψ_i	probability pseudo-random action a_i occurs, scalar
ρ	address space available, scalar
τ	total expected number of transitions before absorption, scalar
θ	weight factor used to combine matrices in error testing, scalar
$\circ$	operator representing element-wise Hadamard product

CHAPTER 1:
Introduction

Despite significant and ever-growing cybersecurity investment [1], cyberspace remains a perilous place. Companies face a nearly one in three chance of experiencing a data breach within the next two years [2], 10% of Americans fall victim to cyber-perpetrated identity theft annually [3], and one in 400 emails is malicious [4]. Cyber attackers hold a significant advantage, in part because they control the time, tempo, and target of their assaults.

During the U.S. National Cyber Leap Year Summit of 2009, authorities touted moving target defense (MTD) as a game-changing cybersecurity concept that would finally reduce this long-held advantage [5]. MTD dynamically alters protected systems to make them less predictable and thus more difficult to attack [5]. A large number of distinct MTD techniques have been cataloged [6] including mutations of system addresses or software, randomization of memory layout and variation of data formats [6]. Each impacts predictability differently, but unfortunately, each can also impact the performance of the defended system. MTD techniques carry overhead like temporary system outages or increases in network traffic load as reconfigurations are deployed [7].

For MTD to achieve truly advantage-shifting results, we need to identify techniques that can amplify the unpredictability of the attack surface [7] without imposing performance-degrading overhead costs [8]. Unfortunately, these are often competing goals [9], as illustrated in Figure 1.1. Defender perspective of MTD ranges from unmanageable wherein reconfigurations are so cumbersome legitimate communications are impossible, through a manageable level of disruption, to the ideal invisible case in which users of the defended system have no awareness reconfigurations are occurring. From the attacker perspective, the changes range from drastic enough to make the attack surface unpredictable, through changes routine enough to be predictable, to the static case in which changes are nonexistent. Implementing MTD is an optimization effort to find the point of balance where an acceptably unpredictable and manageable system is achieved [10].

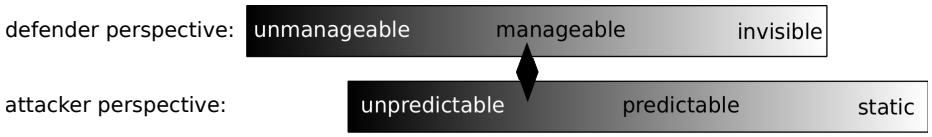

Figure 1.1. Implementing MTD requires trade-off to balance manageability for defenders with unpredictability for attackers

Seeking innovative ways to achieve this balance, we turned to biomimicry, the discipline of finding engineering solutions in nature [11]. We examined predator-prey co-adaptation for relevant strategies because the attacker-defender arms race in cybersecurity creates a similar system [12]. The interplay between bats and moths stood out for particular alignment with aspects of MTD.

Bats first flew between 60 and 95 million years ago, adding an ability to echolocate 50 million years ago [13]. Phased-use of specific amplitude and frequency combinations permits the bat to both scan wide areas and hone in on an individual moth via a terminal buzz [14]. This lethal skill combination enabled bats to dominate predation of nocturnal flying insects until moths developed counter-detection strategies to ensure their own survival [13].

In one such strategy, certain moth species evolved ultrasonic sound detection capabilities [13]. These moths used their awareness of bat echolocation signals to develop graded evasive responses wherein the moth changes its movement depending on the hunting phase of the bat [14]. As illustrated in Figure 1.2, a faintly detected bat can be ignored in favor of continuing other activity. Should a bat in wide area scan be detected, the moth avoids predation by flying out of the area [14]. A bat in the final, terminal buzz phase of the hunt drives the moth to conduct erratic and high-energy maneuvers or dive into leaf-provided acoustic cover [14].

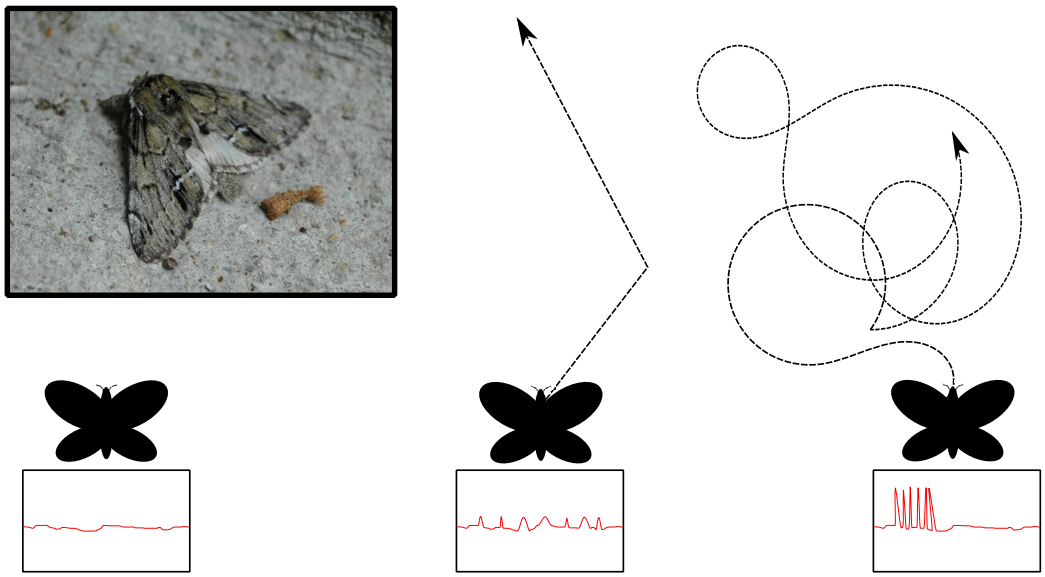

Figure 1.2. Graded evasive response in moths. Inset source: [15].

Graded evasive response has been conclusively observed in thirteen of the nineteen insect families capable of ultrasonic detection, underscoring the clear survival advantage of this strategy [13] and making a compelling case for exploring how graded evasive response might help in MTD. Where the moth wants to avoid the bat for a minimum energy expenditure, the defender similarly wants to avoid the attacker for the minimum performance sacrifice. With the moths in mind, we sought a path to implementing MTD as a graded reaction to cyber attackers so that effectiveness and manageability can both be achieved.

1.1 Objective

The objective of this dissertation is to develop a system capable of simultaneously enhancing the effectiveness of MTD in presenting an unpredictable attack surface and controlling overhead to maintain manageability. As depicted in Figure 1.3, the proposed cyber defense system uses data on attack patterns and system specifications coupled with real-time attack assessment to trigger an optimal defensive action, just as the moth uses instinct and knowledge coupled with detected bat activity to scale evasive flight.

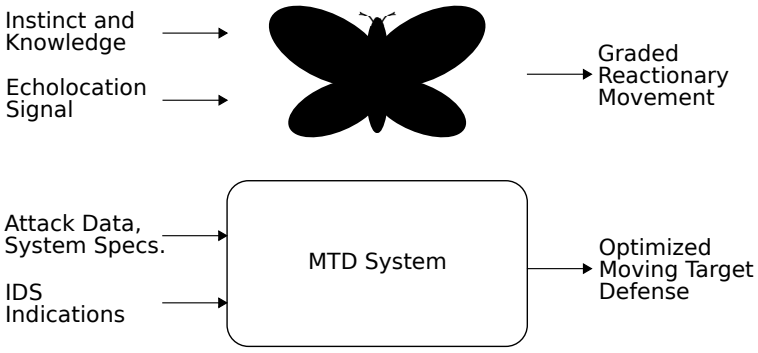

Figure 1.3. Alignment between moth defenses and an optimized MTD scheme

The core mechanism selected to optimally achieve both goals is the partially observable Markov decision process (POMDP). The proposed system, diagrammed in Figure 1.4, includes POMDP formulation, MTD selection, and performance monitoring to identify the optimal action stream $a(t)$ based on the incoming observation stream $\omega(t)$, which represents a partial observation of true attack state $s(t)$.

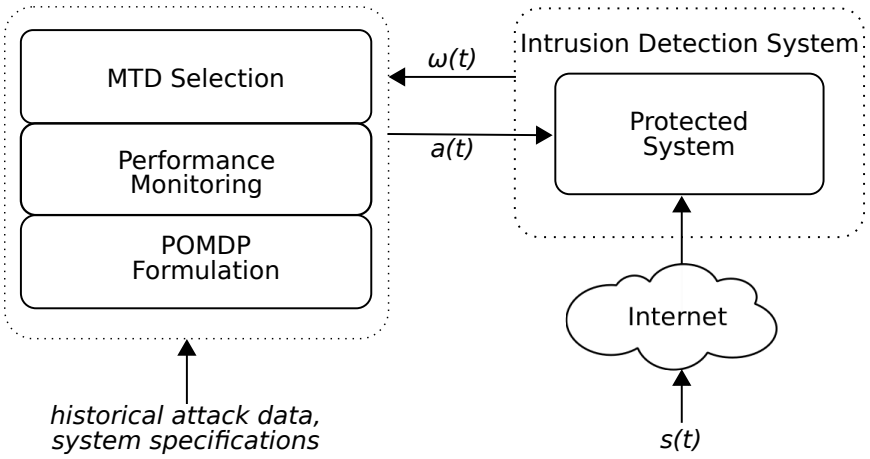

Figure 1.4. The proposed system leverages POMDP to implement cost-controlled MTD.

Such a system is possible because, like the phases of a bat hunt, cyber attacks have distinct phases that occur before an exploit ultimately compromises a device or the attacker goal is otherwise reached [16]. Moreover, such phases require attacker activity that is at least

partially observable by the defender [17], [18] and that follows a pattern that can be distilled from forensics of previous attacks [19]. Using this knowledge, MTD can be triggered to the degree and with the timing required to thwart attacks without the excess overhead that leads to unmanageability.

1.2 Related Work

There are three categories of related work influential in developing our solution toward optimal implementation of MTD. These include generic works related to Markov models, specific cybersecurity applications of POMDP, and finally non-Markovian efforts to optimally employ MTD. This section reviews work from each to highlight where we have adopted and expanded the efforts of other researchers.

1.2.1 Markov Models

To understand the mechanics of absorbing Markov chains, Markov decision process (MDP), and POMDP, we leaned heavily on material from [20], [21] and [22], respectively, extending the fundamentals described in these reference materials to fit the specific context of the envisioned system. The discussion of absorbing Markov chains in [20] was instrumental in developing the predicted performance equations that are critical to the performance monitoring scheme. The descriptions of MDP and POMDP in [21] and [22] confirmed applicability of the model in meeting our design objectives.

While POMDP have been in circulation for decades [23], recent advances make the model more practically adoptable in real-world systems. In particular, online policy development techniques open the door to new applications [24]. In this context, *online* refers to the way in which these techniques use a forward search from the current status of the operational environment, constructing the decision making policy in step with its execution [25]. Offline techniques, on the other hand, construct standing policies covering all possible contingencies so that policy execution is via look-up [25]. Because online techniques construct policies for specific circumstances, rather than for all possible contingencies, online techniques can support systems with much higher dimensionality, and thus complexity, before computational intractability becomes a concern [24].

The two key advances in online policy development fundamental to our proposed system

are partially observable Monte Carlo planning (POMCP) [26] and follow-on determinized sparse partially observable tree planning (DESPOT) [24]. POMCP facilitated scaling up the previous benchmarks in computationally tractable POMDP dimensionality by several orders of magnitude [26]. DESPOT, in turn, improved poor worst-case behavior exhibited by POMCP [24]. As such, our system was built using DESPOT to facilitate action selection from the observation stream and POMDP formulation.

POMDP-based cybersecurity research is generally separated by policy technique. Three use offline solution techniques in developing policies [27]–[29]. This limits the dimensionality the models can incorporate before intractability becomes a concern, though state-of-the-art tools like successive approximations of the reachable space under optimal policies (SARSOP) facilitated significant capacity in the work by Sarraute et al. [28].

1.2.2 POMDP in Cybersecurity

We identified six examples in which POMDP was used to model attack-defense dynamics to improve cybersecurity. A summary of the aspects of each formulation is provided in Table 1.1. The dimensionality columns represent the number of states $|s|$, actions $|a|$, and observations $|\omega|$ that the researchers used in abstracting a cybersecurity system into a POMDP.

Table 1.1. POMDP-based cyber defense systems

Lead Author	Year	Dimensionality			Policy Technique		
		$\|s\|$	$\|a\|$	$\|\omega\|$	Offline	Online	Tool
Kreidl [27]	04	18	3	6	✓		Manual Heuristic Approx.
Sarraute [28]	12	20	5	20	✓		SARSOP
Zonouz [30]	14	64	6	IDS	✓	✓	Value Iteration and Finite Look-ahead
Tipireddy [29]	17	2	3	2	✓		Incremental Pruning
Miehling [31]	18	186	16	8		✓	Modified POMCP
Musman [32]	19	8	4	8		✓	DESPOT

These offline works were influential in understanding cybersecurity model formulation. The

hierarchical approach in [28] was useful in approaching a detailed host-based model, rather than attempting a network-wide model, as the authors demonstrate how host-based models can be combined to track network-wide security [28]. It should be noted that [28] was the only effort intended to optimize attack, rather than defense, but otherwise the structure of the model remains well-aligned with our approach. Influencing the states in our host-based model, Kreidl et al. [27] use a multi-phase attack model similar to the one we eventually adopted in our system. Although MTD was not the focus of [27], the multi-stage attack was attractive for our system because MTD impacts each attack phase differently [33], and a model that represents attack phases is thus necessary to capture this impact. Additionally, the results in [27] support the need for appropriate treatment of incomplete observability in cybersecurity systems, as the authors' improved feedback controller based on the POMDP formulation was able to successfully reject false alarms from the intrusion detection system, whereas a strict rule-based controller was not [27].

The offline approach by Tipireddy et al. [29] was the most reproducible of the field and thus important in gaining experience working with POMDP in a cybersecurity context before formulating our own model. In particular, [29] performed sensitivity analysis across many of the model components, which was helpful in understanding how changes in each component can impact the overall performance of a POMDP-based defensive system.

Online solution techniques as employed by the remaining three works handle much larger dimensionality. In particular, experimentation demonstrated that the modified version of POMCP developed by Miehling et al. [31] generates decisions for models exceeding 100 million states. Both Zonouz et al. [30] and Miehling et al. [31] influenced the way in which we represented attackers in our model formulation. The former validated the ability of a POMDP-based approach to optimally defend against an adversary who omnipotently takes the most harmful adversarial actions [30]. The latter varies attacker type between a finite set of choices which define the attacker skill, aggression, and detectability [31]. Both were useful in understanding how to incorporate the variability of attackers into a single model. Reviewing these efforts drove us to attempt to incorporate the most likely attacker complimented by a mechanism that verifies that attacker is indeed the one being faced.

The study conducted by Musman et al. [32] provides specific tool validation as it compares the autonomous decisions of a DESPOT-controlled cyber response engine to those generated

via SARSOP. In the most extreme case of computational efficiency improvement, the DESPOT-controlled responses matched within 0.8% in attained value for less than 0.03% of the computational time [32]. The scenario was scaled specifically to facilitate this online-offline comparison [32], and as a result, the work was useful as an independent validation of DESPOT in a cybersecurity context before we selected it for use in our proposed system.

Although not POMDP-related, one other work needs to be mentioned because absorbing Markov chains became particularly influential in our work. We found one other instance in which absorbing Markov chain theory was used in a cybersecurity application. Abraham and Nair use an absorbing Markov chain to describe the dynamics of movement through an attack graph [34]. Expected path length and state visits quantify a security metric for the network [34]. We adopt these metrics in an expanded form that incorporates the impact of defensive actions, costs, and partial observability.

Our proposed system builds on and departs from all of these works in key ways. First, the other models do not focus on MTD optimization. Rather, they generally look at when to perform system recovery after attack, rather than when to initiate preventative reconfigurations. Second, we validate our proposed system via an implementation model built via measurements of event probabilities and objective overhead factors from real-world cyber events, while the works surveyed employed expert, yet subjective, rubric-based assessments. Finally, we implement a scheme to continuously monitor how well the POMDP aligns with actual conditions, which was not incorporated into any of the other works.

1.2.3 Optimizing MTD

The research described in three references focuses on non-Markov methods for optimal implementation of MTD, which also relates to and influences our proposed system. This research is summarized in Table 1.2. All three efforts look at techniques in which the overhead expended for MTD can be minimized in balance with maximizing effectiveness.

Two of the references conduct analysis to identify static parameters for system implementation that achieve optimization. Wang et al. [35] use attack traces from distributed denial of service attacks to develop a probability of attack arrival based on historical inter-arrival times. Using a range of values from 1 – 100 to define a generic cost structure, MTD is triggered when benefit outweighs cost based on probability of attack in the near-term [35]. This

Table 1.2. MTD overhead minimization

Lead Author	Year	Trigger	# MTD techniques	Description
DeLoach [10]	14	Random, Reactive	1	Virtual Machine Migration
Wang [35]	16	Random	1	Virtual Machine Migration
Connell [8]	18	Random	2	IP Address, Service Reconfig.

work was an important step toward understanding optimization in the context of realistic attack patterns. Connell et al. [8] determine optimal reconfiguration inter-arrival times for two MTD techniques, which sets their research apart. The work presented in [8] is a specific detailed example of the generic framework for combining MTD techniques explored in [36], and both were instructive in our effort to study optimization when combinations of techniques are employed. Additionally, the overhead measurements in [8] were invaluable in facilitating our validation effort, as these measurements provided an objective cost basis for our validation model and also facilitated comparison between our proposed system and a comparable state-of-the-art random MTD scheme.

The final work explores an intelligent optimization system that triggers address reconfigurations both randomly and in reaction to detected attacker progress and changing system requirements much like the one we propose. DeLoach et al. [10] propose three runtime models to represent system performance constraints, available assets, and vulnerabilities. These models are reasoned over to time a defensive reconfiguration. The research is complimentary to ours in that it supports the use of stochastic models for controlling MTD and explores the impact of uncertainty regarding attack detection on the effectiveness of the system [10]. Our system works similarly but replaces the three runtime models with a single POMDP that facilitates consideration of performance and security factors in a single apparatus. We also use phase-based attack indications rather than an expected arrival rate of attacks for optimal MTD triggering.

1.3 Organization

The remainder of the dissertation details and validates the proposed system and is organized as follows: Chapter 2 provides background information on cyber attacks, MTD, POMDP, and

absorbing Markov chains necessary to describe our proposed system. Chapter 3 describes our solutions approach with details of each subsystem provided in Chapters 4 through 6. Chapter 7 provides validation of the proposed system via a simulated implementation. Chapter 8 concludes the dissertation with a summary of the contributions and avenues for future work. There are four appendices. Appendix A contains a summary of the model used in validation. Appendix B includes additional results from simulated operation of the system. Appendix C describes the simulations conducted including hardware and software employed with snippets of code critical to system operation. Appendix D contains an example of the model specification file used by the proposed system.

CHAPTER 2:
Cybersecurity and Optimization

In this chapter, we cover the background material necessary to describe the proposed system. These topics fall into two areas. The first covers the cybersecurity context in which our system will exist, split into discussions of cyber attack patterns and MTD. The second area encompasses the optimization and modeling concepts used within our system to achieve our design goal of decision making under uncertainty via POMDP and absorbing Markov chains.

2.1 Cyber Attack Patterns

Proactive defense of a system requires an understanding of how attacks unfold. This section provides background information related to trends in cyber attacks and how those trends become predictive tools.

2.1.1 Trends and Statistics

The cyber kill chain describes a seven-step process by which an attacker compromises a target [16]. The attacker moves from reconnaissance, to weaponization, delivery, exploitation, installation, command and control, and finally the ultimate actions on objectives [37]. In wide circulation, this kill chain has been adopted and modified to facilitate various aspects of research in cybersecurity [37]. We adopt a simplified five-phase version in line with [33] because its appears frequently in MTD literature [6], [33], [38], which is our area of interest. The five-phase version includes reconnaissance, access, development, launch, and persistence as depicted in Figure 2.1.

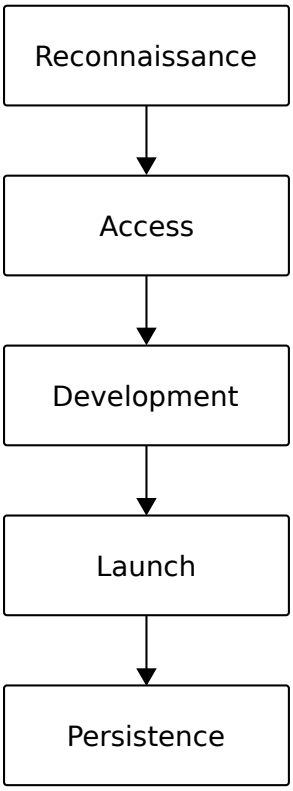

Figure 2.1. The five-phase cyber attack often used in MTD literature. Adapted from [6].

Reconnaissance encompasses the activity the attacker conducts to research, identify, and select a target [37]. For phishing attacks, which comprise as much as 95% of attacks in recent years [39], this is the phase when the attacker gathers the email addresses and related information necessary to make the phishing attempt believable. Such efforts often include crawling internet websites, conference proceedings, social networks and similar resources. Separately, attackers glean technical aspects of potential attack vectors via internet registries, search engines, and other publicly available databases to understand the logical address and domain name space of the target [16].

In the access phase, the attacker collects technical information specific to the target technology necessary to gain access [33]. Social engineering, which involves deceptively manipulating individuals into divulging sensitive information, frequently facilitates accomplishment of this phase [40]. Small scale experiments indicate 50% of people will reveal sensitive

network details over the phone during such social engineering attacks [41]. Attack surface enumeration is also possible via tools designed to rapidly scan large networks for the purposes of network exploration and security auditing [42]. A study from 2011 found that 78% of the traffic inbound to a network was scan related [43]. In a small scale experiment, approximately 50% of the attacks against two honeypots connected to the world wide web were preceded by scan activity of some sort [17]. Such activity can be detected, albeit imperfectly, because it exhibits low packet-per-connection rates [17] or arrives at an unusually large number of ports or devices on a single device or network, respectively [44].

In the development phase, the attacker selects or writes malware and packages it into a deliverable payload. In some cases, attacks are tailored by technically adept adversaries, but malware and exploits that go after wide spread vulnerabilities can also be obtained. Threadkit, the most mentioned exploit kit on the dark web in 2018, can be purchased for just $400 [45]. Such tools remain viable given that vulnerabilities have an average lifespan of seven years [45]. One well understood problem from the common vulnerabilities and exposures database made the Recorded Future top ten list of most exploited vulnerabilities for three consecutive years, long after security officials were made aware of the problem [45]. With patching inconsistent at the host-level, even the release of software patches can offer attackers a cookbook for conducting attacks against un-patched systems [46].

The launch phase covers the transmission of the weapon to the target. In the case of advanced persistent threat actors, data from the Lockheed Martin Computer Incident Response Team collected from 2004-2010 indicates that the most prevalent delivery vectors for weaponized payloads are email attachments, websites, and removable media [16]. Despite security education, the majority of corporate data breaches are caused by human error in one of these categories [47]. To ease delivery, client application data files such as Adobe portable document format or Microsoft Office documents often serve as the weaponized deliverable so that unsuspecting human users will help circumvent system security measures [16]. Email attacks circumvent enterprise security by luring privileged users into downloading and executing malware [39]. Website delivery vectors often abuse the domain name system by registering a name that resembles a well-known entity or editing legitimate records to include malicious subdomains, both of which fool victims into downloading malware themselves [48]. As for removable media, 29 different peripheral device types capable of facilitating malware delivery were identified as of 2017, ranging from flash drives to

keyboards [49]. Flash drives are particularly noteworthy. In a 2016 experiment, researchers found that 98% of the flash drives dropped on a college campus were picked up and at least 45% of them were subsequently connected to a device, with victims' altruism and/or curiosity driving them to both plug the drives in and open the contained files, usually without taking any security precautions [50].

Once the delivered malware lands at the target system, it may or may not succeed in compromising the device. Both the target defenses and the expertise with which the attack was crafted will impact likelihood of success. Assuming that it succeeds, if persistent access is desired, the attacker will take steps in the final persistence phase to ensure such access is maintained via remote access trojans or other backdoor installations [16]. Data exfiltration is a typical goal of this phase [16]. The adversary may also compromise a given host for intermediate use permitting lateral motion to more lucrative victims [16]. Anomalous activity on the protected system can indicate attainment of the persistence phase, with detection methods relating to unusual computational load, network traffic volume, network traffic destination, and privilege escalation examples of the many in circulation [51].

There are many variations of activity per phase, with each differing in likelihood of occurrence, success, and detection. Making sense of these compound possibilities to take preventative action is the work of attack modeling.

2.1.2 Attack Models

Attack models translate the many events and statistics measured in forensic analysis of cybersecurity events into probabilities useful in predicting, and ultimately preventing, the next attack. These models first took their now familiar shape describing progress of an attacker toward an objective in the attack graphs of the late 1980s [34]. As researchers began to quantify multitudes of cybersecurity influences, stochastic processes became the dominant technique by which they described attack progressions, especially when analysis of event likelihood was desired [52].

The goals of stochastic attack models are varied. Certainly, research aims to improve intrusion detection of in-progress attacks [53]–[56], but more proactive themes are also in circulation [57]–[60]. For example, in [57], the authors combine stochastic modeling with analysis of overarching patterns in cyber attacks to make recommendations regarding

prioritization of security efforts.

In building a model for either purpose, there are a variety of ways of defining the set of system states s and the set of system observations ω. The alert sequence of an intrusion detection system (IDS) is a common template for defining states [54], [58]. Other options include command sequence [53] and vulnerability life cycle [61]. The state of individual hosts can combine to offer a summary of network state as a whole [60]. Another alternative for network-wide analysis is to define s in terms of network nodes, particularly useful when the goal of the model is to identify the probable next attack victim given the nodes currently under attack [56]. Other authors have defined states in terms of the strings of sequential exploits en route to an ultimate attack goal within a network regardless of physical location [31], [34], [62]–[64].

Because of computational disadvantages in manipulating and analyzing models with large state space, simplified models that combine all alerts into a smaller set of states aligned with attack phases have proven useful [29], [55], [58]–[60], [65]. The chain can be completely collapsed to just two states, *good* or *compromised* [29] or expanded to include intermediate states along the lines of the cyber kill chain [8]. The observation set ω is often compressed as well, from a direct feed of IDS alerts to some subset representing the results of initial analysis, reducing the set of possible observations [55]. Even if $s \equiv \omega$, retaining both permits consideration of missed or false detection [54].

With states and observations defined, the probabilities of transitioning from any state or observation to another are calculated from historical analysis of past attacks, either from global or local data. Once compiled, the probabilities that events in the attack model (1) occur and (2) are detected are used to put together metrics including attack occurrence and timing [62]. These metrics can be used to assess security adequacy of a system and take action to correct deficiencies [34].

2.2 Moving Target Defense

MTD corrects such deficiencies by initiating system changes that reduce the likelihood of attack success by reducing the predictability of the attack surface [5]. MTD techniques are classified based on the layer at which the reconfiguration occurs within five categories as follows: dynamic data, dynamic software, dynamic run time environment, dynamic platform,

and dynamic network [33]. Dynamic techniques also vary in the phase of attack they are most likely to impact as summarized in Table 2.1. No single technique thus cataloged is effective against all five attack phases, and most are effective against only one or two [6].

Table 2.1. Impact of dynamic techniques on attacks by phase. Adapted from: [33].

MTD	Attack Phase				
category	Reconnaissance	Access	Development	Launch	Persistence
Network	✓			✓	
Data			✓	✓	
Software			✓	✓	
Platform		✓	✓		✓
Runtime environment			✓	✓	

Dynamic network techniques change network properties and configurations to thwart attackers and are particularly effective during the reconnaissance and launch attack phases [33]. As a well known example of network MTD, in internet protocol (IP) hopping, network addresses are regularly shuffled to ensure the knowledge gained during the attacker network reconnaissance is perishable. By doing so, the defender prevents the attacker from controlling the tempo of attacks as actions must be taken before information becomes obsolete [7]. Such reconfiguration also disrupts standard techniques for flow and session isolation, which decreases the likelihood a successful attack can be mounted [7]. Address mutation shows particular promise toward deterring and deceiving reconnaissance-based attacks like mathematically propagated worms while offering the added benefit of increasing the detectability of such attacks [66]. Another technique, random route mutation, uses dynamic routing to thwart adversarial eavesdropping and certain denial of service attacks and has been shown in simulation to reduce attacked or disrupted traffic to just 10% of its pre-implementation levels [67]. The 2018 survey of techniques cataloged nineteen unique examples of this type of dynamic movement [6].

Under dynamic data techniques, data is dynamically modified in storage or presentation to complicate the development of exploit payloads capable of harvesting data of interest [33].

In one patented implementation, protected data is partitioned into multiple files that are obfuscated via encryption and then moved around a computer network or remote cloud-based environment [68]. As another example, relatively simple diversity in data formatting as depicted in Table 2.2, though easy for the human brain to interpret, complicates data exfiltration by making format-based extraction less likely to succeed [33]. Eight unique examples of this type of MTD have been cataloged [6].

Table 2.2. Data presented differently to complicate exfiltration. Source: [33].

Format 1	**Format 2**
<Age = 23;	<ID=00132573;
Gender = Male;	Gender=M;
ID = 132573;	Salary=75K;
Salary = $75000;>	Age=10111;>

Dynamic software also impacts the development and launch phases and comprises techniques that maintain program functionality while modifying program instruction sequences, internal data structures, and other previously static qualities so that the internal software state is no longer deterministic relative to input [33]. For example, software diversity on a given network involves deploying multiple versions of an executable, each version holding different vulnerabilities, to minimize the spreading of an attack exploiting any one of them [6]. Although some portion of the individual hosts may still fall victim, the overall network is better protected [6]. One of the more interesting conclusions found in researching this example technique was that even diversity of the diversity implementation itself was important in maintaining the defensive advantage as any predictability in how software versions were deployed eased attacker progress [69]. Thirteen unique examples of this type of dynamic movement are currently in circulation [6].

In the category of dynamic platform, changes are made to the operating system, virtual machine instance, or other low-level environmental factors [33]. For example, ESCAPE is a proposed MTD architecture that migrates docker containers around different hosts to complicate attack [70]. In simulation, ESCAPE improved the survival probability for a single targeted container moved around 20 hosts to above 90% in a scenario where, without

ESCAPE in place, the survival probability was less than 5%. Eighteen unique examples of dynamic platform exist [6]. These techniques impact the access and development attack phases [6].

Dynamic runtime environment efforts impact the development and launch phases. The 31 cataloged examples [6] seek to create memory address space randomization and instruction set randomization to thwart exploitation of software vulnerabilities [33]. As an exemplar, buffer overflow attacks are thwarted when address space layout permutation complicates attacker ability to predict where a pointer will land in the heap or stack [6].

In addition to these discussions of the effectiveness of MTD, overhead is also a mechanism by which techniques can be classified. There are various factors at play: initial investment costs to obtain or modify hardware or software; investment in human capital to ensure sufficient expertise is on hand to implement and operate; and per-event overhead related to system downtime, memory use, or network traffic incurred each time the reconfiguration is triggered [6]. Often, these factors are assessed on a rubric because of a lack of detail in available technique specifications [6].

Developing metrics of MTD effectiveness [38], evaluating composability [38], and quantifying overhead [8] are critical and active fields of research, necessary before MTD matures beyond individual craft techniques to a formally understood scientific discipline [38].

2.3 Decision Making Under Uncertainty via POMDP

MTD implementation involves a series of decisions related to timing, degree, and detail of defense employment. The system we propose requires a mechanism for automating and optimizing such decisions, and MDP offers such a mechanism [21]. Given that attacker progress and intention are difficult to perfectly detect or discern, Markov-based models suited for problems with state uncertainty become particularly attractive. In fact, hidden Markov models are a dominant tool applied in more than 30% of discrete cybersecurity forecasting models [65].

For systems requiring handling for both sequential decision making and persistent state uncertainty, POMDPs emerge as the dominant technique applied [71]. POMDPs are best known in the field of operations research where they have been in circulation for

decades [23]. POMDPs have gained wider traction recently via new solution techniques that accommodate high dimensionality [72]. In this case, *dimensionality* refers to the product of the quantities of states, actions, and observations incorporated in the model.

POMDP can be thought of as a description of the process diagrammed in Figure 2.2. The process has two inputs: time-series of states $s(t) = s_j, s_j \in s$, a discrete random variable; and time-series of actions $a(t) = a_i, a_i \in a$, which is user selected and thus deterministic.

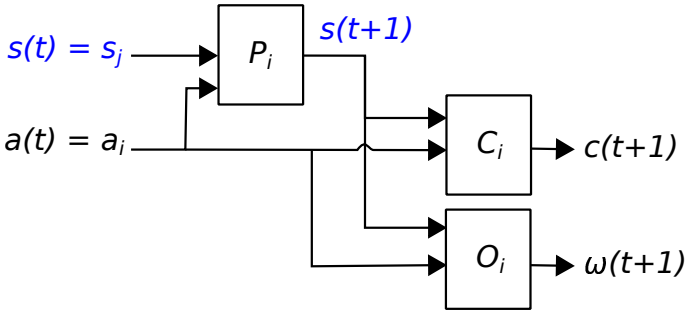

Figure 2.2. In this functional diagram of a POMDP, given a random starting state $s(t)$ and user selected action $a(t)$, the components P_i and O_i of the POMDP describe the probability with which $s(t+1)$ and $\omega(t+1)$ occur conditioned on $a(t) = a_i$, while component C_i dictates $c(t+1)$, the deterministic cost incurred in this time step. Values in blue are hidden from the user.

The output of the first stage is $s(t+1)$, which is a discrete random variable drawn from the state transition probability defined as

$$\Pr[s(t+1) = s_k | s(t) = s_j, a(t) = a_i] = p_{i,j,k} \tag{2.1}$$

where the notation $Pr[x = y|z]$ represents the probability that $x = y$ conditioned on z. The set of possible values for $p_{i,j,k}$ are contained in state transition probability matrix P_i of the form

$$P_i = \begin{bmatrix} p_{i,1,1} & p_{i,1,2} & \cdots & p_{i,1,n} \\ p_{i,2,1} & p_{i,2,2} & \cdots & p_{i,2,n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{i,n,1} & p_{i,n,2} & \cdots & p_{i,n,n} \end{bmatrix}.$$

Any single row j of P_i represents the probability mass function describing the state transition probabilities from s_j conditioned on action a_i, and thus $\sum_{k=1}^{n} p_{i,j,k} = 1$.

The true state $s(t)$ is hidden from the user for all t. Instead, the user receives partial information about $s(t+1)$ via the observation $\omega(t+1)$, which is a discrete random variable that behaves according to

$$\Pr[\omega(t+1) = \omega_k | s(t+1) = s_j, a(t) = a_i] = o_{i,j,k} \tag{2.2}$$

where the set of all possible values for $o_{i,j,k}$ are contained in observation probability matrix O_i. Matrix O_i is of the form

$$O_i = \begin{bmatrix} o_{i,1,1} & o_{i,1,2} & \dots & o_{i,1,n} \\ o_{i,2,1} & o_{i,2,2} & \dots & o_{i,2,n} \\ \vdots & \vdots & \ddots & \vdots \\ o_{i,n,1} & o_{i,n,2} & \dots & o_{i,n,n} \end{bmatrix}$$

where any single row k of O_i is a probability mass function describing the likelihood of an observation occurrence in s_j conditioned on action a_i. It follows that $\sum_{k=1}^{n} o_{i,j,k} = 1$.

Additionally, $s(t+1)$ and $a(t)$ result in deterministic output $c(t+1)$ according to the cost matrix C_i defined as

$$C_i = \begin{bmatrix} c_{i,1,1} & c_{i,1,2} & \dots & c_{i,1,n} \\ c_{i,2,1} & c_{i,2,2} & \dots & c_{i,2,n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{i,n,1} & c_{i,n,2} & \dots & c_{i,n,n} \end{bmatrix}$$

where $c_{i,j,k}$ is the cost incurred for transitioning between states s_j and s_k under action a_i.

The components of a POMDP are listed in Table 2.3. The n possible system states are represented by state vector s while the m available actions are contained in action vector a. Partial observations of state are drawn from a set of k options in ω. The system stochastically moves among states according to the conditional probabilities in transition probability matrix P_i with elements $p_{i,j,k}$ following Equation 2.1. Rewards or costs are generated by this state transition according to C_i where $c_{i,j,k}$ represents the reward (+) or cost (−) of taking action a_i in s_j and landing in s_k. Observations occur as a result of state change according

to the conditional probabilities in O_i with elements $o_{i,k,m}$ following Equation 2.2. The final component γ is used to balance future and immediate rewards, with $\gamma = 0$ putting complete emphasis on immediate rewards, and $\gamma = 1$ representing no discount such that rewards retain their full value across all horizons.

Table 2.3. Generic POMDP component formulation

Component	**Description**	**Size**
state vector s	all possible system states	$1 \times n$
action vector a	available actions	$1 \times m$
observation vector ω	all possible observations	$1 \times k$
state transition probability matrix P_i	likelihood system transitions between any two states under a_i, combining m P_i together, $P = \{P_1, P_2, ..., P_m\}$	$n \times n$
cost matrix C_i	cost $(-)$ or reward $(+)$ for moving between states under a_i, combining m C_i together, $C = \{C_1, C_2, ..., C_m\}$	$n \times n$
observation probability matrix O_i	likelihood of observation by state under action a_i, combining m O_i together, $O = \{O_1, O_2, ..., O_m\}$	$n \times k$
discount factor γ	factor balancing immediate and future rewards, $0 \leq \gamma \leq 1$	scalar

To use a POMDP to facilitate optimal decision making, an agent is required that can be decomposed into two sub-agents: (1) a state estimator and (2) a policy [22].

2.3.1 State Estimation

The state estimator uses the model to create a belief state $b(t)$ representing the estimate of current state given $\omega(t)$ and $a(t)$. Depending on the dimensionality of the POMDP, this can

be done either via a Bayesian belief update or via particle methods [24]. Either way, $b(t)$ expresses the complete state-observation-action history in a compact form the policy can use for decision making.

When the Bayesian belief update is employed, belief $b(t)$ is a vector of length n. The vector is a probability mass function describing system state and takes the form

$$b(t) = [b_1(t), b_2(t), ..., b_n(t)]$$

where $b_j(t)$ is the probability that the system is in state s_j at time t. Each member of the vector is obtained recursively via

$$b_j(t) = \xi o_{i,j,m} \sum_{k=1}^{n} p_{i,k,j} b_k(t-1) \tag{2.3}$$

where $\omega(t) = \omega_m$, $a(t) = a_i$ and

$$\xi = \left(\sum_{j=1}^{n} \left[o_{i,j,m} \sum_{k=1}^{n} p_{i,k,j} b_k(t-1) \right] \right)^{-1}$$

to ensure $\sum_{j=1}^{n} b_j(t) = 1$ in accordance with the law of total probability [24].

Belief $b(t)$ enables the state estimator to maintain a more accurate assessment of true state than would be afforded based on observations alone, which in turn facilitates higher quality decision making. For example, we found that using $\arg\max[b(t)]$ to estimate state rather than $\omega(t)$ consistently improved the accuracy of state estimate by approximately $u = 29\%$. The improvement in state estimate u was calculated as

$$u = \left(\frac{1 - \Pr[\omega(t) = s(t)]}{1 - p_D} - \frac{1 - \Pr[\arg\max[b(t)] = s(t)]}{1 - p_D} \right) \times 100\%$$

where probability of detection p_D describes the likelihood that state and observation agree. We explored the value of u in simulation over the range of $0.76 \leq p_D \leq 1.00$ for the proposed system under validation conditions that will be described in Chapter 7. A comparison of the accuracy of $\arg\max[b(t)]$ and $\omega(t)$ from the same simulation is depicted in Figure 2.3, where $\Pr[\arg\max[b(t)] = s(t)] > \Pr[\omega(t) = s(t)]$ across all test cases.

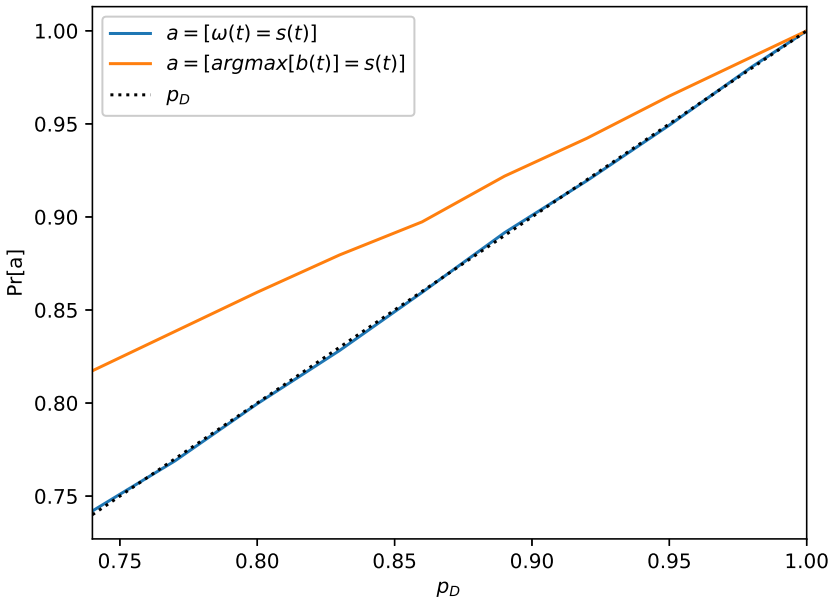

Figure 2.3. The belief state $b(t)$ maintains a more accurate estimate of true state than $\omega(t)$.

For systems of significant complexity, $b(t)$ changes from a probability mass function to a particle representation of state estimate updated via a particle filter [24]. Our proposed system will employ a direct Bayesian update in accordance with Equation 2.3, but particle techniques are considered a path for future expansion of the complexity the proposed system can incorporate.

2.3.2 Policy

The policy describes the action that maximizes expected rewards or minimizes expected costs given $b(t)$. Under complete observability such that $s \equiv \omega$ and O_i is an identity matrix for $1 \leq i \leq m$, the system collapses to an MDP wherein a static policy Π is represented as a vector of size $1 \times n$ indicating the appropriate action to take in each state. The value V_Π of the policy is found according to

$$V_\Pi = \beta(0)(I_n - \gamma P_\Pi)^{-1} C_\Pi$$

where $b(0)$ is the vector of initial state probabilities, and P_Π and C_Π are matrices built from P and C such that $p_{\pi,i,j} = p_{\Pi(i),i,j}$ and $c_{\pi,i,j} = c_{\Pi(i),i,j}$ [21]. The optimal policy can be found by searching through all m^n possible permutations of state-action pairing to identify the pairing that maximizes V. Search efficiency is gained via dynamic programming techniques including value and policy iteration [21].

For POMDP, expected value must be projected over the range of belief states, rather than discrete state-action pairings. The efficient offline options for developing a static $b(t) \leftrightarrow a(t)$ mapping for problems with low dimensionality are well summarized by [22] and [73].

For problems with high dimensionality, online solution techniques are needed [26]. The "curses" of dimensionality and history render the policy otherwise intractable [24]. To ensure relevance in complex real-world applications, the proposed system employs online solution techniques.

In particular, online solutions of the POMCP family are well suited [31]. Although not the first algorithm to extend Monte-Carlo planning to POMDP [74], POMCP uses a tree search process to efficiently update both the policy decision and the belief state enabling effective and efficient decisions in POMDP that were previously computationally intractable [26]. For example, for a model with approximately 250,000 finite states, POMCP achieved the same performance, measured in average reward earned, as the state-of-the-art offline technique but used just four seconds of online computation as compared to the 1,000 seconds required by the offline method [26].

Building on the success of POMCP, new algorithms further increase efficiency by reducing the number of simulations while still achieving near-optimal decisions. One such advancement is DESPOT, which reduces the number of simulations necessary to achieve optimal decision by randomly selecting a subset of all possible future trajectories [24]. The algorithm has been published via [24] and also made available as part of the Approximate POMDP Planning Toolkit [75]. In experiments, DESPOT has been shown to improve upon the worst-case behavior of POMCP while providing comparably rapid and optimal results [24].

As illustrated in Figure 2.4, when incoming observations are received and translated into $b(t)$, simulations are conducted to explore cost incurred in various futures. The volume of simulations is controlled by a particle count q, which each simulation projecting one possible

future to a user-selected horizon d. Each horizon is described by a randomly selected action and observation where the action is drawn from a with each member having an equal likelihood, while the observation is drawn from ω according to P and O and the selected action. If each horizon considered the m available actions and k possible observations, the complete simulation tree would contain $m^d n^d$ nodes [24]. To improve computational performance, DESPOT explores a determinized sparse subset of the nodes in the complete tree as prescribed by particle count q that the user selects based on a tolerance for sub-optimal decision making [24]. When collected together, the set of q simulations overlay into an approximation of the complete tree with some observations removed [24].

In a single simulation, after d is reached, the costs encountered at each horizon are added to the previous horizon discounted by γ, propagating all the way back to the root. As such, the scalar cost incurred in a given simulation c' is defined as

$$c' = \sum_{i=1}^{d} \gamma^i c_{a'(i),\omega'(i-1),\omega'(i)} \tag{2.4}$$

where the subscripts of $c_{i,j,k}$ are annotated by the $'$ symbol to clarify that these are the actions and observations from this specific simulation history, rather than from the observation stream or action selection occurring in the operational environment.

The next action $a(t)$ is identified via the branch with the lowest expected cost when all q simulation costs are added together weighted by likelihood. This is why the illustration uses a heavy dot to represent $a(t)$ selection occurring one horizon beneath the root node. The simulation tree building process is repeated after each new observation.

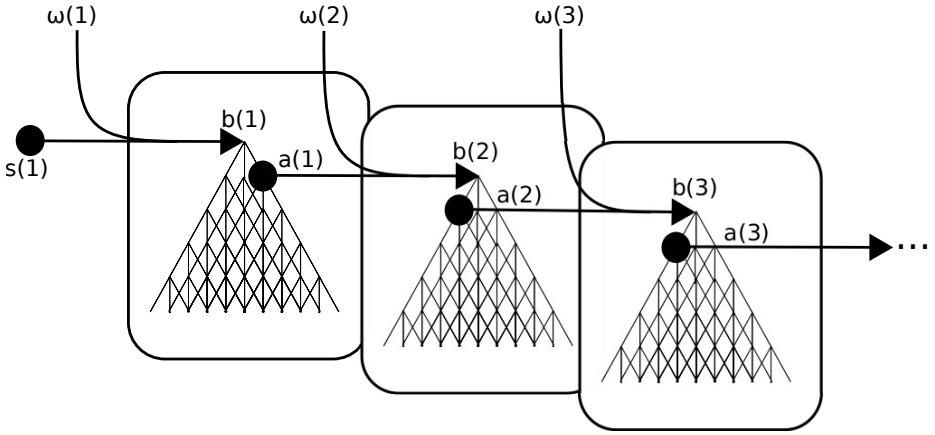

Figure 2.4. With an online POMDP policy, once incoming observation $\omega(t)$ is received and used to update $b(t)$, simulations are conducted to identify the optimal action $a(t)$ that minimizes expected cost.

The key parameters for DESPOT decision quality are particle count q and depth d. Perfect optimality in the infinite horizon is achieved as $q \to \infty$, $d \to \infty$, but sufficient optimality is possible at much more manageable values [24].

2.4 Absorbing Markov Chains

Our model uses an absorbing Markov chain to model cyber attacks. This section outlines notation and properties of absorbing Markov chains useful in describing the proposed system. In an absorbing Markov chain, there is at least one *absorbing* state s_i from which the system cannot exit, i.e., $\Pr[s(t+1) = s_i | s(t) = s_i] = 1.0$. The remaining states are called *transient* states.

In modeling cyber attacks, we model attacker goal state s_n as an absorbing state for two reasons. First, this state represents the ultimate attacker goal, which once attained remains satisfied until intervention by the defender guarantees removal of the attacker to restore the system [34]. Such attack recovery efforts are variable and manual processes that can take weeks to resolve [76], putting them on a different time scale than the attacks represented in our model, which occur in matter of minutes [17]. We also found that adopting an absorbing state improved the ability of the proposed system to work as designed in ramping up defensive efforts as attack becomes imminent. This aspect will be described in Chapter

3.

The fundamentals of absorbing Markov chains described in [20] are restated here in the notation adopted throughout this dissertation. The standard form of P_i, a single member of P, for a Markov chain with one absorbing state and $n-1$ transient states is

$$P_i = \begin{bmatrix} Q_i & R_{n-1} \\ \overline{0}_{n-1} & 1 \end{bmatrix} \tag{2.5}$$

where Q_i is the $(n-1) \times (n-1)$ upper left partition of P_i, partition R_{n-1} is the $(n-1) \times 1$ vector of absorption probabilities by starting state taken from column n of P_i, and $\overline{0}_{n-1}$ is a $1 \times (n-1)$ zero vector [20].

The fundamental matrix N of an absorbing Markov chain is defined as

$$N = (I_{n-1} - Q_i)^{-1} \tag{2.6}$$

and contains the expected number of transient state visits before absorption under a_i, with $n_{i,j}$ indicating the expected visits to s_j if the system starts in s_i [20]. The matrix I_{n-1} is an identity matrix of size $(n-1) \times (n-1)$.

Given $b(0)$, a $1 \times (n)$ vector of initial state probabilities, we define $\beta(0)$ as the first $n-1$ elements of $b(0)$, i.e., the partition describing initial transient state probabilities. The expected transitions before absorption τ follow as

$$\tau = \beta(0) N_i \zeta \tag{2.7}$$

where ζ is an $(n-1) \times 1$ vector of ones [20].

The matrix N_2 contains the variances for each values in N and can be found from N as

$$N_2 = N \left(2N_{dg} - I_{n-1}\right) - N_{sq} \tag{2.8}$$

where $N_{dg} = N \circ I_{n-1}$ and $N_{sq} = N \circ N$ [20]. The operator $\circ$ represents the Hadamard or element-wise product.

Thus, many of the properties of system operation are available via analysis of P_i. To make

such analysis possible for the proposed system, we extend these formulations to include weighting factors for P and Q that account for impacts of partial observability and the history of actions. These extensions are described in Chapters 3 and 6.

2.5 Summary

This chapter summarized the background material necessary for describing our proposed system and the ways in which it implements manageable, effective MTD. With this scaffolding in place, the next chapters describe our proposed system.

CHAPTER 3:
POMDP-based Moving Target Defense Scheme

We propose a novel method of harnessing the advantages of moving target defense while controlling the overhead. Our moth-inspired approach scales MTD in proportion to the proximity of the threat. The method employs less costly defense available in early attack phases and ramps up to more expensive options when justified by imminent attack accomplishment.

The proposed system takes advantage of the following facts and assumptions that were supported in Chapter 2. First, cyber attacks have distinct phases that build up to an exploit ultimately compromising a device [16]. Reaching each phase requires attacker activity that is at least partially observable by the defender [51]. Also, for the family of MTD techniques that carry a per-event overhead, reducing the frequency of reconfiguration occurrences reduces the overall overhead expended for defense and thus makes MTD more manageable.

The structure of the problem facilitates leveraging POMDP to combine all influences into a single framework that identifies the optimal MTD at a given point in time to achieve attack suppression for the lowest long-term cost. This chapter details the model at the core of this solution and offers an overview of how the proposed system employs that model.

3.1 POMDP Model Specification

The POMDP model is the core of the proposed system. Thus, the description of the system must begin by outlining how the cyber attack-defense process is abstracted into the model. This section introduces a POMDP formulation that supports MTD optimization. The generic model components introduced in Chapter 2 are now defined within an MTD context as summarized in Table 3.1. POMDP was selected because it facilitates optimal decision making even under persistent state uncertainty [71], which will be the case for cyber defense applications in which attacker progress is not perfectly known by the defender.

Table 3.1. POMDP formulation for model of MTD dynamics

Component	**Description**	**Size**
state vector s	list of attack phases	$1 \times n$
action vector a	list of MTD	$1 \times m$
observation vector ω	list of IDS indications ($s = \omega$)	$1 \times n$
transition probability matrix P_i	likelihood that system transitions between attack phases under MTD a_i,combining m P_i together, $P = \{P_1, P_2, ..., P_m\}$	$n \times n$
cost matrix C_i	overhead ($-$) incurred for moving between any two phases under MTD a_i, combining m C_i together, $C = \{C_1, C_2, ..., C_m\}$	$n \times n$
observation probability matrix O_i	likelihood IDS indication aligns with attack phase under MTD a_i, combining m O_i together, $O = \{O_1, O_2, ..., O_m\}$	$n \times n$
discount factor γ	factor balancing immediate defensive overhead with long term attack penalties, $0 \leq \gamma \leq 1$	scalar

The state vector s is a discrete list of the n phases of a cyber attack as

$$s = [s_1, s_2, ..., s_n].$$

State s_1 is the earliest phase of the attack, and s_n represents the ultimate attack goal. The intermediate states represent incremental progress toward s_n and can be described in a multitude of ways including those described in Section 2.1.1. Although intermediate phases of the attack may be skipped, phases are ordered from 1 to n such that the imminence of attack can be inferred by state index.

The action set a is the discrete list of m available MTD as

$$a = [a_1, a_2, ..., a_m].$$

The first in the list, a_1, represents the *nil* option in which the defender takes no action at all. The remaining $m-1$ options are what the defender has available to thwart the attacker. It can be helpful, though not necessary, to order the defenses by either effectiveness in thwarting the attacker or overhead incurred for use. We have kept to this convention, so that a_m is the most effective defense available to the defender.

The observation set ω is the discrete list of possible observations that could be received from the upstream IDS. For the proposed system, the burden of processing myriad attack indicators lies with the IDS such that possible observations are drawn from the possible attack phases, i.e., $s \equiv \omega$. As such, there are n members in ω.

The state transition probability set P contains a set of m state transition probability matrices, with individual member P_i describing the state transition probabilities under MTD a_i. The matrix component $p_{i,j,k}$ describes the probability that the attack progresses to s_k from s_j if MTD a_i is employed. Each P_i proscribes a Markov chain unto itself as shown in Figure 3.1. Given the convention we adopt in which a_1 describes the *nil* option, P_1 describes the pattern of unencumbered attacks.

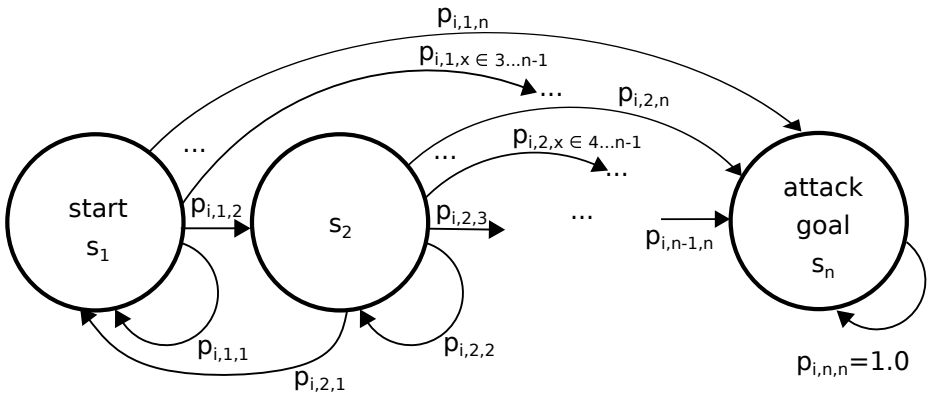

Figure 3.1. A Markov chain with transition probability matrix P_i describes the system dynamics under action a_i.

The cost set C contains m cost matrices, with cost matrix C_i describing the overhead incurred

under MTD a_i. Individual matrix component $c_{i,j,k}$ is the specific cost incurred if the system transitions from s_j to s_k under MTD a_i. To facilitate optimization as intended, the costs must represent both attack penalties and the overhead of MTD. The attack penalty c_{atk} is prevented so long as $s(t) \neq s_n$. MTD overhead is treated as independent of attack phase such that it can be considered a constant of c_i for $s_j \in s, s_k \in s$. Thus any individual element $c_{i,j,k}$ can be found as

$$c_{i,j,k} = \begin{cases} c_i & j < n \\ c_i + c_{atk} & j = n \end{cases}.$$

We define c_{def} as a $1 \times m$ vector of defensive costs c_i to succinctly describe the MTD overhead by itself.

The defender has partial observability of attack phase. The characteristics of this partial observability are captured in observation set O that contains m observation probability matrices denoted by O_i that contain the likelihood of alignment between $\omega(t)$ and $s(t)$. We introduce the general form of O to facilitate consideration of the way individual defenses may improve or degrade the ability to discern true system state but restrict our work to consider defense and observation processes independent such that $O_i = O_j$ for $1 \leq i, j \leq m$.

Finally, a discount factor γ is required to establish the desired balance between immediate and future costs. This scalar value will fall between 0 and 1. Lower values place more emphasis on immediate costs, with future costs becoming increasingly influential proportional to γ until $\gamma = 1$ when all costs carry the same emphasis [22].

Together, the components $\{s, a, \omega, P, C, O, \gamma\}$ form the POMDP integral to our proposed system. The goal of formulating the attack-defense dynamics in this way is to facilitate cost exploration. From the model, the user can identify the optimal $a(t)$.

3.2 Absorbing State to Represent Attack Goal

In the early stages of developing this system, we envisioned a model capable of representing the complete dynamics of the attack-defense interaction, all the way through recovery. Recovery is represented in all but one of the POMDP-based works discussed in Chapter 2. For example, recovery is represented by a stochastic return to an *uncompromised* state following initiation of a *reset* action in [32]. We found, however, that including the recovery

process within the POMDP was problematic for two reasons.

First, recovery is difficult, if not impossible, to accurately represent as a stochastic process. Once an exploit lands on a device, recovery involves a variable manual process dependent on the nature and extent of the damage. It can take weeks to fully recover [76]. Second, in initial exploration of online solution techniques for POMDP, we found that inclusion of the recovery process in the model led to undesirable a_1 (*nil*) action selection in the high-risk states closest to s_n. Just when defenses were needed most, they failed to deploy.

To overcome both of these issues, we adopt an absorbing state at s_n such that

$$p_{i,n,n} = 1.0 \text{ for } 1 \leq i \leq m.$$

An illustration of the influence of the absorbing state is presented in Figure 3.2. These box plots contain the projected costs of all possible discounted futures when either a nil or *def* action is taken in the next time step. In both scenarios (a) and (b), the system is currently in the penultimate state s_{n-1}. Also, in both scenarios, $p_{2,(1,\ldots,n-1),1} = 1.0$ such that selecting action a_2*:def* returns the system to safety. In scenario (a), the model includes recovery such that $p_{i,n,1} = 1.0$. In scenario (b), attack recovery is not possible as attack is an absorbing sate, i.e., $p_{i,n,n} = 1.0$.

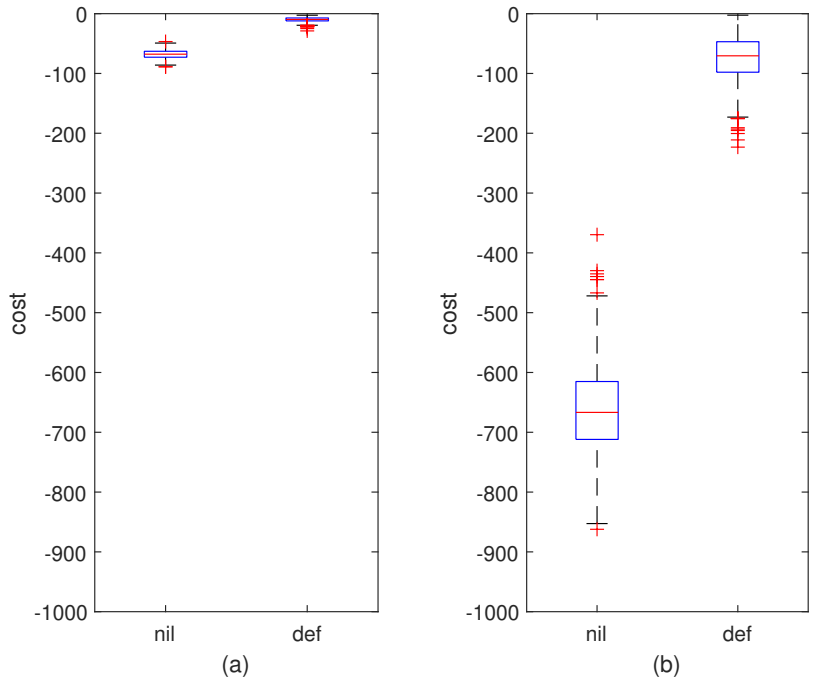

Figure 3.2. In an illustrative case with recovery in place (a), the attack penalty and defensive costs are within 60 cost units. Under identical parameters other than the addition of the absorbing state (b), the distance increases ten-fold, reducing the difficulty of discerning the optimal action.

The optimal decision in either scenario is the one that minimizes cost, keeping it closest to zero. In both cases, *def* is the better choice, indicated by the *def* box plots being closest to zero, but the wider spread between *nil* and *def* costs in scenario (b) sets better conditions for the simulation-based approach used in online POMDP solution techniques to select the optimal decision.

This POMDP formulation with the absorbing end state is similar to the goal attack states in [31], with a key difference: in our model, revisiting lower numbered states is permissible from any state except s_n while the model in [31] assumes monotonicity in that even intermediate phases toward attack accomplishment cannot be reverted once achieved. Because our system is focused on optimizing MTD, the monotonicity assumption no longer holds [10]. MTD introduces uncertainty for attackers with the intent of forcing them to revisit earlier attack phases. Thus, developing a model incorporating potential for backward state transi-

tions right up until the system enters s_n is an important contribution toward model-based control for MTD.

3.3 System Overview

The proposed system, illustrated in Figure 3.3, uses the incoming observation stream $\omega(t)$ to produce an outgoing MTD stream $a(t)$ matched to the assessed threat. For the family of MTD techniques that carry a per-event overhead such that reducing the frequency of reconfiguration occurrences reduces the overhead expended for defense, the proposed system thus sets the conditions for optimal system defense wherein attacks are thwarted for the minimal overhead.

Within the system, the necessary tasks to produce $a(t)$ are divided between three subsystems. These subsystems are the POMDP formulation subsystem, the MTD selection subsystem, and the performance monitoring scheme. Each is explored in more detail in Chapters 4 through 6, but the following sections introduce each and describe how the subsystems interact.

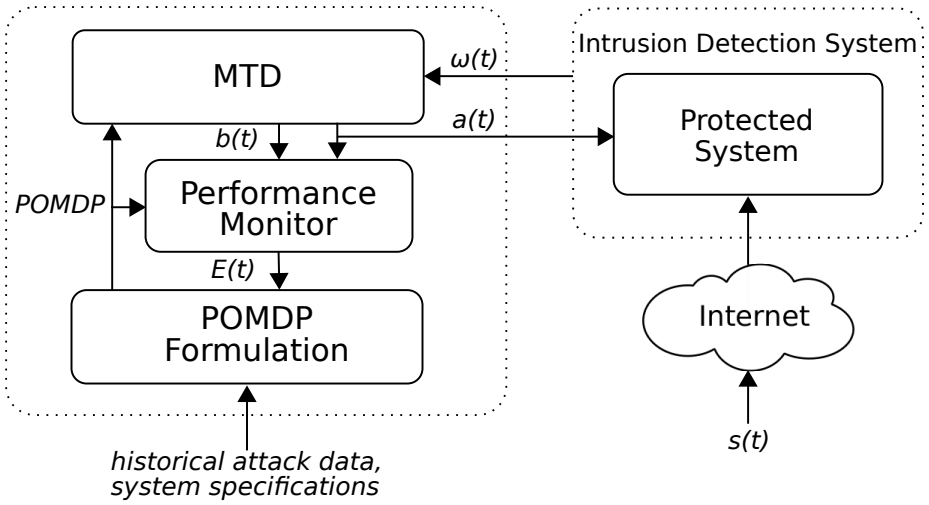

Figure 3.3. The proposed MTD scheme determines the optimal action to take each time an observation is received based on a POMDP formulated from training data. The POMDP is reformulated when performance metrics vary from the values expected.

3.3.1 POMDP Formulation Subsystem

At initialization, the POMDP formulation subsystem quantifies the values of each component of the POMDP described in Section 3.1. Together, $\{s, a, \omega, P, C, O, \gamma\}$ represent the stochastic interplay between the attacker and defender. Overall system performance is highly dependent on the accuracy of the POMDP in terms of how closely aligned with the actual attack-defense dynamics it represents. As such, POMDP formulation requires historical data on cyber attacks, activity trends for the defended system, and technical specification of each available defense.

Attack analysis, defense analysis, prioritization of competing requirements, and an assessment of the upstream IDS turn the available data into $\{s, a, \omega, P, C, O, \gamma\}$. The most useful formulation is tailored to the context in which the proposed system exists so that the resultant POMDP captures the most relevant threats, options, and priorities.

Once the components have been developed, the POMDP is used by both the MTD subsystem and the performance monitoring scheme. The MTD subsystem employs the model to optimally implement MTD while the performance monitoring scheme uses it to trigger reformulation when required. In preparation for reformulation, the acquisition of historical data and other supporting information should be considered a continuous processes that also leverages knowledge gained via operation of the proposed system. More details on the POMDP formulation system are provided in Chapter 4.

3.3.2 MTD Selection Subsystem

The MTD selection subsystem uses the POMDP and incoming partial observation stream $\omega(t)$ to identify the optimal action $a(t)$. Although the notation is aligned with a sense of time, $\omega(t)$ may be time- or event-driven, but $a(t)$ is always event-driven in that a new action is selected every time a new observation is received.

Action selection is via the DESPOT algorithm described in Chapter 2, which conducts simulations to identify the MTD leading to the lowest expected discounted cost. Because the system is built under the assumption that the IDS is imperfect, and thus $\omega(t)$ is only a partial observation of true state $s(t)$, decisions rely on belief state $b(t)$ derived via a Bayesian update of the previous belief state $b(t-1)$ via Equation 2.3.

In this way, actions remain near-optimal even as flawed observations are received. Based on the cost-benefit analysis achieved via DESPOT, early phases of attack are met with low or no cost actions. As the attacker progresses toward s_n, more significant defensive overhead becomes justified to match more imminent attack risk. More detail regarding the MTD subsystem is provided in Chapter 5.

3.3.3 Performance Monitoring Scheme

The performance of the proposed system is entirely dependent on the effectiveness with which we have abstracted true attack-defense dynamics in the operational environment into the POMDP model. For brevity, we will refer to this consideration as *model effectiveness* throughout this dissertation. If the model is ineffective, the system will fail to initiate defenses optimally. At best, defenses will be used more frequently than necessary, causing excessive overhead. At worst, the attacker will have an easier time pushing the system into s_n.

Because model effectiveness is so important, the proposed system includes a performance monitoring scheme that flags anomalous system trends to facilitate corrective action via POMDP reformulation. The scheme tracks f_ω, the frequency with which each member of ω is received; f_a, the frequency with which each member of a is initiated; and χ, the average per-decision overhead. These measured values are compared to what would be expected under perfect model effectiveness. Details of how expected metrics f_ω^*, f_a^* and χ^* are developed from $\{s, a, \omega, P, C, O, \gamma\}$ are provided in Chapter 6.

The expected metrics (indicated by superscript *) are transformed into error quantities by tracking the average distance between actual and expected values. The error metric ϵ is composed of these distances and contains three components. The first component, ϵ_ω, is quantified as

$$\epsilon_\omega = \frac{1}{n} \sum_{i=1}^{n-1} \left(f_\omega(i) - f_\omega^*(i) \right)^2 . \tag{3.1}$$

Error in f_a, ϵ_a, is quantified as

$$\epsilon_a = \frac{1}{m} \sum_{i=1}^{m} \left(f_a(i) - f_a^*(i) \right)^2 . \tag{3.2}$$

Error in χ, ϵ_χ is quantified as

$$\epsilon_\chi = (\chi - \chi^*)^2 \,. \tag{3.3}$$

Collectively, these errors form $\epsilon = [\epsilon_\omega, \epsilon_a, \epsilon_\chi]$, a 1×3 vector of error metrics. The errors ϵ_ω, ϵ_a, and ϵ_χ are compared to error thresholds, respectively, represented as a 1×3 vector $\delta = [\delta_\omega, \delta_a, \delta_\chi]$. POMDP reformulation is triggered via error signal $E(t)$, which is defined as

$$E(t) = \begin{cases} 1 & \epsilon > \delta \\ 0 & else \end{cases}$$

wherein reformulation begins when $E(t) = 1$.

This scheme is critical to attaining and maintaining confidence in the system as proposed because the attack-defense interplay is ever evolving. The monitor of model effectiveness facilitates rapid corrective action to ensure the model-based system remains useful. Reformulation is also indicated when the system enters s_n to understand the nature of the successful attack to improve performance before resuming operation.

3.4 System Performance Metrics

Quantifying the value of the proposed system requires two metrics: average per-decision overhead, χ, and attack suppression ϕ. We define average per-decision defensive overhead χ as the sum of the incurred overhead averaged over the number of decisions that have been made at step t as

$$\chi = \frac{1}{t} \sum_{x=0}^{t} c_i(x)$$

where $c_i(x)$ is the vector containing the overhead incurred by the system by the decision at step x.

We define attack suppression ϕ of the system by comparing τ, the expected number of state transitions before the system enters s_n, to the expected number of transitions before s_n if no defenses are in place. The actual value τ comes from observing the system in action, while the undefended value is found from static analysis of the attack dynamics described by P_1.

Thus, we label the undefended number τ_1, which can be found

$$\tau_1 = \beta(0)(I_{n-1} - Q_1)^{-1}\zeta$$

where Q_1, $\beta(0)$, I_{n-1}, and ζ are defined according to the convention analysis of absorbing Markov chains introduced in Chapter 2. If the defended system is attacked at step τ, we would have expected $x = \frac{\tau}{\tau_1}$ attacks to have occurred by step τ in the undefended system. Thus, the proposed system has suppressed $x - 1$ attacks. As a proportion, attack suppression ϕ follows as

$$\phi = \frac{x - 1}{x}.$$

By substituting $x = \frac{\tau}{\tau_1}$, we express ϕ directly from τ as

$$\phi = 1 - \frac{\tau_1}{\tau}.$$

We generally multiply by ϕ by 100% to express as a percentage. These two metrics will be applied in Chapter 7 to validate the utility of the proposed system.

3.5 Summary

This chapter introduced our solution approach for implementing MTD that is both effective and manageable. We propose a system that uses partial observations of attack phase to optimize MTD selection and timing, reducing the overall number of reconfigurations without reducing attack suppression capability. The next chapters explore each subsystem in more detail.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4:
POMDP Formulation

The first step in implementing the model-based optimization scheme proposed is formulating the model at its backbone. This chapter explores the steps of POMDP formulation introduced in Chapter 3. Model formulation occurs at initialization and re-occurs as required by the performance monitoring scheme.

We selected POMDP because it accommodates all facets of attack-defense dynamics including the probability with which MTD thwarts attacks by phase and the persistent state uncertainty related to incomplete observation of attacker activity and intention. The POMDP components $\{s, a, \omega, P, C, O, \gamma\}$ are defined as listed in Table 3.1.

These components are derived from manual analysis of all available data related to attack, operation, and defense of the protected system. Examples include forensics from attacks against this and other systems, system specifications and requirements, and traffic analysis of the protected system. Deriving the components can be broken up into four channels of analysis as illustrated in Figure 4.1. The next sections give more insight into each channel.

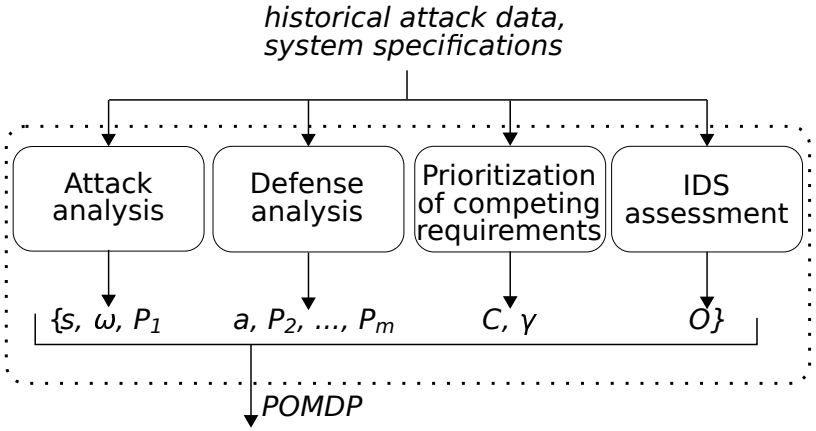

Figure 4.1. POMDP formulation subsystem

4.1 Attack Analysis

The goal of attack analysis is to identify s, ω, and P_1 conducive to achieving MTD optimization. Cyber attacks occur in discrete stages as discussed in Chapter 2. In our model, these attack stages become the states of the Markov chain illustrated in Figure 3.1. While the model can have as few as two states, achieving our stated design goals requires inclusion of the intermediate attack phases that MTD is designed to impact. Thus, sources for selecting the states of a model include both attack forensics and MTD specifications.

The next step is to determine a transition basis for the system in question. Event-based examples include per-connection, per-session, or per-IDS-alert. Time-based transitions are also possible.

Next, we leverage attack forensics to define the probabilities with which an attacker moves between states. To be most powerful, these probabilities should be derived from forensic analysis of attacks against similar targets, but that information can be hard to obtain. In its absence, the insight available from ethical hackers and expert cybersecurity practitioners must be leveraged. Examples of the types of published resources useful for this process include cyber threat intelligence reports such as those from Mandiant [77], cybersecurity industry white papers like those sponsored by International Business Machine [2] or Verizon [78], U.S. government accountability reports [3], and academic research papers [17].

Generically, the Markov chain is fully connected to capture the attacker ability to skip, loiter in, and revisit states. All together, the probabilities are gathered in P_1, the first member of P. The only exception to full connectivity is s_n, as the final state is absorbing, i.e., $p_{1,n,n} = 1.0$, in accordance with the discussion in Chapter 3. The other probabilities are estimated from occurrence counts in data of past attacks against similar devices. We step through an example of this process in Chapter 7.

In our system, the observation set $\omega \equiv s$. The upstream intrusion detection system carries the burden of collating various indications of attack activity into an assessed attack phase. Thus, once s is determined, ω is as well. With s, ω, and P_1 determined, attack analysis is complete.

4.2 Defense Analysis

The next step in POMDP formulation is an assessment of the available defenses to determine a, $P_{2,\ldots,m}$, and c_{def}. The catalog of available defenses constitute a, a $1 \times m$ vector, keeping in mind that the first of these defenses is a non-action labeled the *nil* defense. The transition probabilities under defense a_i are represented as P_i.

The specific values in P_i for $1 < i \leq m$ require careful consideration of the phase-impact of defense a_i. The literature offers the starting point, as surveys of MTD including [6] provide qualitative considerations of the phase-impact of more than 90 MTD techniques. Translating these broad assessments to a specific transition probability requires consideration of the technical descriptions of the defense in context of the attack model developed during attack analysis.

It is most useful to define P_i as a function of P_1. This dependency permits rapid update of P after any change in P_1 so that the system can stay in step as attack patterns change. Many MTD techniques follow the general form of alternating some facet of the system among a set of choices. Thus, the reconfiguration involves rotating some facet of the device among k versions. A version may be re-selected in the reconfiguration process such that there is probability of failure $p_f = \frac{1}{k}$ wherein no change occurs from the attacker perspective and thus the defense fails to impede progress. The complimentary probability $p_s = \frac{k-1}{k}$ represents the likelihood that the defense succeeds. For an MTD that returns the system to s_1 with probability of success p_s, the remaining transition probabilities can be found as $p_f \times P_1$. This process of developing P_i is depicted in Figure 4.2 and must be repeated for each member of a.

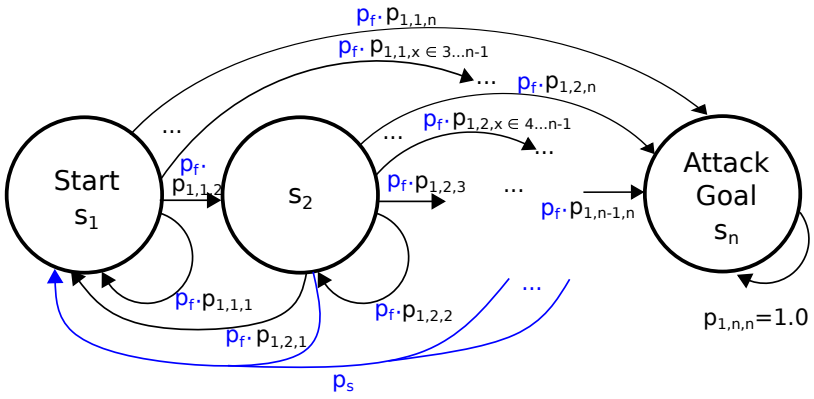

Figure 4.2. An MTD a_i that returns the system to s_1 with p_s is represented by a Markov chain with transition probabilities P_i defined in reference to P_1.

Defense analysis also involves quantifying the overhead c_i incurred when a_i occurs. In particular, the overhead must relate to the goals of the optimization effort. For example, if the optimization goal is to suppress attacks while minimizing downtime of services, C_i should reflect the downtime for deploying reconfiguration a_i. In general, one-time overhead expenses such as those required for installation do not translate into the proposed approach and must be accounted for separately. When multiple cost factors matter for the optimization goals, they must be scaled and combined into a single value that reflects the prioritization of each. For the purposes of describing the proposed system, we discuss cost in terms of availability because it offers a tangible and objective basis for comparison. Thus, c_i carries a unit of seconds to express the down time incurred when a_i is deployed. Costs are imposed as negative values so that the most expensive defense carries the minimum c_i. The defense analysis concludes once P_i and c_i have been developed for every member in a.

4.3 Prioritization of Attack Prevention

Next, we translate tolerance for attack risk and defensive overhead into the model by formulating C to reflect the relative prioritization between these competing goals. The cost set C contains the m matrices, with matrix C_i indicating the total overhead incurred if action a_i is taken in a given state, resulting in landing in another state.

In the most generic case, C_i is of size $n \times n$ to reflect the differences in cost when taking a_i in any state and landing in any another. Few defensive costs are state-dependent, so in

practice C_i frequently contains repeated entries of scalar action cost c_i with the exception of the final row, which is where the attack penalty c_{atk} is added to c_i to penalize the system for entering s_n.

As discussed in Chapter 2, POMDP-based systems achieve optimization by identifying the action that will maximize discounted expected rewards (or minimize discounted expected costs) in the infinite horizon. Costs are imposed as negative values such that the most expensive defense corresponds with the minimum value in cost vector c_{def}. Thus, c_{atk} is defined relative to this defense as

$$c_{atk} = \nu \min[c_{def}] \tag{4.1}$$

in which we define ν as the attack penalty scaling factor. The prioritization of attack suppression over the optimization factors is thus directly proportional to ν. The defender must find ν such that attacks are thwarted at an acceptable cost.

This balance is achieved by considering the tipping points in ν that result in changes in the optimal policy of an MDP-equivalent system. MDP systems can be solved for optimal policies via dynamic programming techniques as described in Chapter 2. As ν increases, preventing attack becomes increasingly influential in minimizing the cost of a given policy until it becomes the only influential factor.

At that point, the system will conduct expensive but effective reconfigurations regardless of attack phase. Meanwhile, as ν decreases, the defensive costs become more influential in the overall value determination until ν is so low that defenses are skipped in the interest of minimizing cost even when attack is imminent. Analysis of optimal MDP policies over a range of ν values will identify up to $n \times m$ points for which ν drives a change in policy as more expensive but effective defenses are used in increasingly more attack stages as ν increases.

These shifts are illustrated for a generic system in Figure 4.3, which contains a plot of the states with given action applied as a function of attack penalty scaling factor ν. Under the assumption that cost and effectiveness are directly proportional such that the defense with the least overhead is also the least adept at thwarting attacks, at $\nu = 0$, there is no attack penalty applied, and thus the most inexpensive action is applied in all n states. At

the other end of the spectrum when $\nu = \infty$, attack is weighted so heavily that the action with the most significant cost and effectiveness is applied in all n states. Given the interest in optimization prerequisite for considering the proposed system, the defender seeks to identify ν in the mid-range of values such that the most expensive defenses are used only in cases of imminent attack, true to the moth-based inspiration.

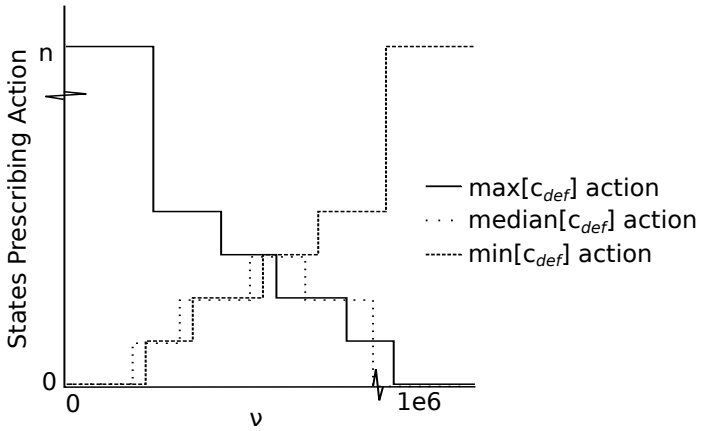

Figure 4.3. As ν increases, the proportion of states using the most aggressive defenses increases until the tipping point when aggressive, repeated defense overrides any effort to implement optimization.

The discount factor γ influences the priority of immediate versus future rewards, with the minimum value 0 resulting in future overhead being ignored, and maximum value 1 considering all horizons of equal importance. We have conducted our analysis at values in the range $0.75 \leq \gamma \leq 0.95$ to reflect the importance of attack suppression while still imposing cost control measures.

4.4 IDS Assessment

The final component of POMDP formulation involves assessment of the upstream IDS to determine the probability of detection p_D, i.e., the probability that $\omega(t)$ and $s(t)$ align. From there, assessment must be made to determine where the error, $1 - p_D$ falls by state. These values are collected into the observation probability matrix O_i, which is $n \times n$ in size and reflects the probability of receiving observation x in state y conditioned on action a_i as $o_{i,x,y}$.

With the IDS upstream of the proposed system, the IDS performance specifications begin the assessment that will end at O. The p_D as well as the probabilities of false alarm p_{FA} and missed detection p_M must be determined, usually from tests of the IDS under conditions in which attack activity is well understood. While there are many uncertainty conditions $\{U_1, U_2, U_3, ...\}$ possible, our work explores the performance of the proposed system under three conditions of the form

$$O_i = \begin{bmatrix} p_D & p_{FA} & \dots & 0 & 0 \\ p_M & p_D & p_{FA} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \vdots & p_M & p_D & p_{FA} \\ 0 & 0 & \dots & p_M & p_D \end{bmatrix} \tag{4.2}$$

in which p_{FA} and p_M are restricted to the immediate neighboring states.

4.5 Summary

This chapter explored the four assessments that must occur during POMDP formulation. The complete model $\{s, a, \omega, P, C, O, \gamma\}$ is relayed to both the performance monitoring scheme and the MTD subsystem. The next two chapters will detail how each of these subsystems employ the model to achieve the overall objective of overhead minimization for an MTD scheme.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:
Moving Target Defense Selection

This chapter describes the processes that occur within the MTD selection subsystem. When an observation is received, the system leverages the POMDP to simulate the impact of each possible action to determine that which is most beneficial. A detailed view of this subprocess is shown in Figure 5.1. The sections in this chapter describe the Bayesian belief update and DESPOT processes in more detail.

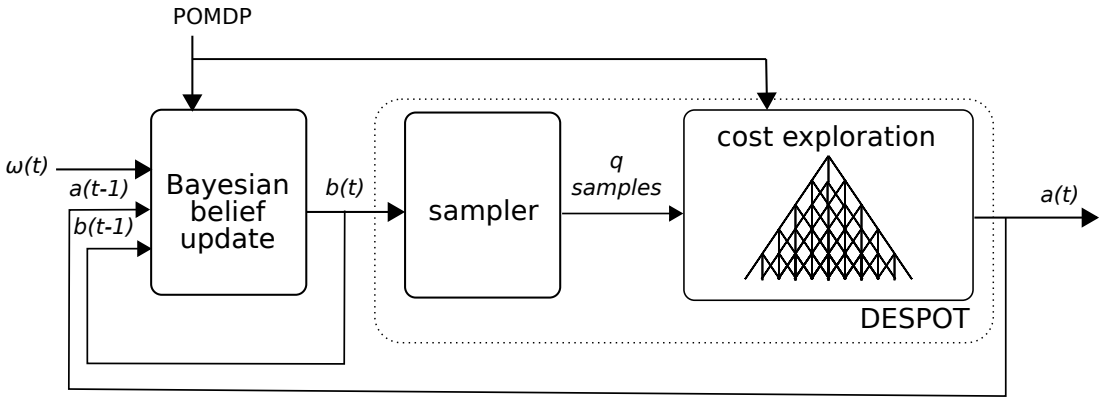

Figure 5.1. MTD selection subsystem. This process occurs every time an observation is received and results in an optimal action being taken.

Because the state of the system is only partially observable, the system takes action based on the belief state vector $b(t)$ rather than the true state $s(t)$. Vector $b(t)$ is developed via Bayesian belief update in accordance with Equation 2.3, and element $b_i(t)$ is the probability that the system is in state s_i at time t. Thus, $b(t)$ represents attack phase as a probability mass function over s. At initialization, $b(0) = [1, 0, 0, ..., 0]$ to represent the assumption that the system is not yet compromised when the cyber defense system is implemented.

5.1 Control Parameters

Once $b(t)$ is determined, the next step uses the DESPOT algorithm to identify the optimal action. As described in Chapter 2, DESPOT is both a software [75] and algorithm [24] that

uses simulation to identify the optimal defensive action. This section describes the selection of the control parameters for DESPOT as well as its role in selecting the next action.

The key parameters that control the quality of DESPOT decisions are particle volume q and simulation depth d as a sufficient number of trees must be explored to a sufficient horizon to ensure optimality is achieved [24].

Identifying these parameters is an optimization effort in itself, particularly to select d such that the priorities of optimization are enforced. The undefended system dynamics are used to do so because of the compounding attack penalties that occur once the system reaches s_n. The key system metric toward determining d is τ_1, the expected number of state transitions before the system is compromised if no defensive action is taken, found from Q_1, the partition of P_1 as defined in Equation 2.5. The steps before absorption without defense τ_1 follow Equation 2.7 and fundamental matrix absent defense N_1 follows Equation 2.6.

To facilitate discussion, the collection of all simulations exploring for the optimal action can be thought of as a tree, with each individual action-observation simulation history flowing from the tree root out to a branch. Within this visualization, the *nil* branch describes the branch of the tree with a series of d repeated selections of a_1 have been implemented. This branch is expected to enter s_n at horizon τ_1, which results in the system suffering from the attack penalty, regardless of action, for every subsequent horizon. The expected cost c incurred at each node following entry to s_n is thus c_{atk}.

To determine the expected cost of the next decision, the expected cost at each subsequent node is propagated back down the tree toward the root, discounted by γ before being added to the next horizon in accordance with Equation 2.4. For the *nil* branch, by definition

$$c_{a(i),\omega(i-1),\omega'(i)} = \begin{cases} 0 & i < d \\ c_{atk} & i \geq d \end{cases}$$

such that

$$c'_{nil} = \begin{cases} 0 & d < \tau_1 \\ \gamma^{\tau_1 - 1} \cdot c_{atk} \sum_{\alpha=0}^{d-\tau_1} \gamma^{\alpha} & d \geq \tau_1. \end{cases} \tag{5.1}$$

Only the a_1 sequence is expected to be attacked by τ_1, as defenses a_2 through a_m are more

effective at thwarting attacks than a_1. Thus, we can assume no other sampled scenarios include the attack penalty by horizon τ_1 and thus the most significant expected cost of any other simulation branch c'_{min} reflects sequential use of the most expensive reconfiguration such that $c_{a(i),\omega(i-1),\omega'(i)} = \min[c_{def}]$ for $1 \leq i \leq d$, which when substituted into Equation 2.4 leads to an expected branch cost of

$$c'_{min} = \min[c_{def}] \sum_{\alpha=0}^{d} \gamma^{\alpha} \tag{5.2}$$

where $\min[c_{def}]$ represents the value in vector c_{def} with the greatest magnitude, as all costs are imposed as negative values. By convention, this is also the cost of action a_m.

When $c'_{nil} \geq c'_{min}$, a_1 is not viable even in low numbered states, which subverts the cost minimization design goal for this system. Thus, we seek to select control parameter d to prevent this condition from occurring.

Beginning from desired endstate $c'_{nil} > c'_{min}$, and substituting in the definitions of either value from Equations 5.1 and 5.2, we find that

$$\nu\gamma^{\tau_1-1} \min[c_{def}] \sum_{\alpha=0}^{d-\tau_1} \gamma^{\alpha} < \min[c_{def}] \sum_{\alpha=0}^{d} \gamma^{\alpha}.$$

Dividing either side by $\min[c_{def}]$, a negative value, it follows that

$$\nu\gamma^{\tau_1-1} \sum_{\alpha=0}^{d-\tau_1} \gamma^{\alpha} > \sum_{\alpha=0}^{d} \gamma^{\alpha}.$$

Geometric series of the form $\sum_{i=0}^{n-1} r^k = \frac{1-r^n}{1-r}$ when $r \neq 1$ [79]. Replacing the summations on either side with their equivalent expressions following this form and multiplying either side by $(1-\gamma)$, we find that

$$\nu\gamma^{\tau_1-1}(1-\gamma^{d-\tau_1+1}) > (1-\gamma^{d+1})$$

so long as $0 \leq \gamma < 1$. Distributing the ν term on the left side before grouping all γ terms

together on the left, it follows that

$$\nu\gamma^{\tau_1-1} - \nu\gamma^{d} + \gamma^{d+1} > 1.$$

Recall that attack penalty scaling factor ν must be significant enough to express the attack risk tolerance of the defender. In practice it is usually many orders of magnitude greater than γ such that this expression can be expressed with significant terms only as

$$\nu\gamma^{\tau_1-1} > \nu\gamma^{d}.$$

Taking the logarithm of either side after dividing by ν, we can see a boundary for parameter d exists at $d < \tau_1 - 1$. Abiding by this expression will help achieve the desired cost minimization by keeping a_1 viable in low-risk states.

With d selected, selection of q is next. In theory, the error between the value of the true optimal policy and the policy derived via DESPOT becomes arbitrarily small as $q \to \infty$ [24]. However, computation time to exhaust all q scenarios through a depth of d increases with q, so in practice q is selected to achieve quality decisions within an acceptable time frame, which is dependent on the hardware employed by the system. We discuss the determination of this value for the specific hardware used in simulating the proposed system in Chapter 7.

5.2 Action Selection

With the control parameters established, DESPOT identifies the optimal MTD response to incoming belief state $b(t)$, which is ingested into DESPOT as part of an extensible markup language file that also includes full specification of POMDP components $\{s, a, \omega, P, C, O, \gamma\}$. A sample file is included as Appendix D.

DESPOT translates $b(t)$ into representative set of 256 particles from which q particles are selected as the first node, or root, in the trees DESPOT will explore. Each of the q scenarios are overlaid into a single tree that explores the costs of action-sequences projected from the initial node out to a length of d. The trees explore every possible action sequence but only a subset of the possible observation sequences to improve the computational efficiency with which DESPOT selects the next defense [24].

Since the goal of this exploration is to identify the optimal action, costs are averaged by likelihood over all scenarios that contain a_i as the next action. These costs will include the overhead of the defenses used in the scenario as well as the attack penalty for those scenarios in which ω_n occurs. The next action $a(t)$ is selected as that which has the lowest expected discounted cost. Once selected, the action is implemented and also relayed to the Bayesian belief update module and performance monitoring scheme.

As an example, we explore this process for a system with $n = 2$, $m = 2$, $q = 6$, and $d = 1$ where $b(t) = [0.5, 0.5]$. In practice, parameters n, m, q, and d are significantly larger, but this small illustration is useful in tracing the decision process. The trees used for exploration are illustrated in Figure 5.2. In this case, there are $2 \times 2 \times 2 = 8$ possible scenarios, but given that $q = 6$, scenarios $s_2a_1s_1$ and $s_2a_2s_1$ are not included in consideration. In this particular simplistic case, elimination of these scenarios is ideal as $p_{1,2,1} = p_{2,2,1} = 0.0$ and thus neither is possible. There are a variety of other paths to determining the subset of scenarios selected that leverage boundary and worst-case scenario analysis as defined in [24].

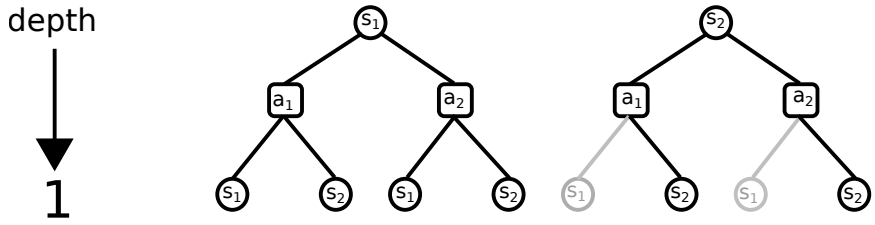

Figure 5.2. DESPOT trees for determining next action when $b(t) = [0.5, 0.5]$ in the proposed system given $n = 2$, $m = 2$, $q = 6$, and $d = 1$.

The expected cost incurred at each of the eight nodes is compiled in Table 5.1 to illustrate how the MTD with the lowest expected overall cost is identified. In this case, we follow the convention of the proposed system in that c_{atk} is only incurred in $s_n = s_2$ and $c_1 = 0$. Also, the absorbing state dictates that $p_{i,2,2} = 1.0$ for all i.

Table 5.1. Projected costs of each scenario in Figure 5.2.

scenario	progression	c	Pr[scenario]
1	$s_1 a_1 s_1$	0	$b_1 p_{1,1,1}$
2	$s_1 a_1 s_2$	c_{atk}	$b_1 p_{1,1,2}$
3	$s_1 a_2 s_1$	c_2	$b_1 p_{2,1,1}$
4	$s_1 a_2 s_2$	$c_2 + c_{atk}$	$b_1 p_{2,1,2}$
5	$s_2 a_1 s_2$	c_{atk}	$b_2 p_{1,2,2}$
6	$s_2 a_2 s_2$	$c_2 + c_{atk}$	$b_2 p_{2,2,2}$

The expected cost of taking a_1 can be defined by conditional expectation

$$\mathrm{E}[c|a_1] = b_1 \left(0 \cdot p_{1,1,1} + c_{atk} \cdot p_{1,1,2}\right) + b_2 \left(c_{atk} \cdot p_{1,2,2}\right)$$

which simplifies to $\mathrm{E}[c|a_1] = 0.5 c_{atk} \left(p_{1,1,2} + 1\right)$. The other option carries an expected cost of

$$\mathrm{E}[c|a_2] = b_1(c_2 \cdot p_{2,1,1} + (c_2 + c_{atk}) \cdot p_{2,1,2}) + b_2 \left((c_{atk} + c_2) \cdot p_{1,2,2}\right)$$

which simplifies to $\mathrm{E}[c|a_2] = 0.5 c_2 \left(p_{2,1,1} + c_{atk} p_{2,1,2}\right) + 0.5(c_{atk} + c_2)$. The proposed system select the action with the largest expected value, which is also the value closest to 0 given that both defensive overhead and attack penalties are expressed in the cost matrices in C as negative values.

5.3 Summary

Received observations are translated into optimal defensive action by the MTD selection subsystem. These results are also relayed to the performance monitoring scheme to ensure the POMDP remains an accurate and thus effective tool for defense optimization. This chapter explained the details of the MTD selection subsystem. The next chapter describes the performance monitoring scheme.

CHAPTER 6:
Performance Monitoring Scheme

This chapter details the performance monitoring scheme that triggers POMDP reformulation when there are indications of poor model effectiveness or completed attack. Recall that we define model effectiveness as the measure of how well true operational conditions have been abstracted into the model. The subsystem is illustrated in Figure 6.1. The following sections describe how the expected performance metrics (indicated by superscript *) are obtained and how threshold vector δ are identified for comparing these expected metrics to the actual values.

Expected metrics frequency of observation f_ω^*, frequency of action f_a^*, and overhead per action χ^* are ideal for two reasons. First, measuring the actual values is a straightforward process as all three metrics are available to the defender. Second, all three metrics stabilize to consistent values shortly after system initialization. Not all metrics support real-time tracking of performance, with the most obvious negative example being attack suppression. Although an important metric, attack suppression cannot be accurately measured until after an attack occurs, making it unsuitable for triggering early intervention to correct poor system performance.

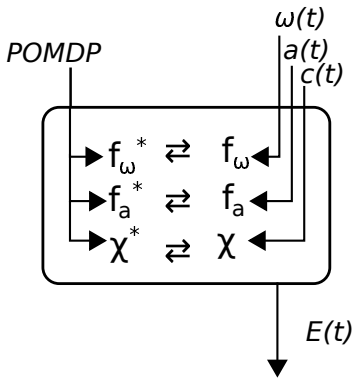

Figure 6.1. The performance monitoring scheme compares real-time metrics to expected values and sets $E(t) = 1$ when POMDP reformulation is required.

6.1 Expected System Performance Metrics

The system requires expected values of each metric under perfect model effectiveness. It is possible to predict these metrics by applying absorbing Markov chain theory and POMDP theory under the assumptions and within the structure of our model. These assumptions and structural qualities include a single absorbing state s_n, 1:1 correspondence between state and observation spaces ($s \equiv \omega$), defensive overhead that is independent of state, and neighboring uncertainty as defined in Section 4.4. We developed these equations with support from empirical analysis of each quality in simulated operation of the system, which assisted in identifying how to combine absorbing Markov chain and POMDP theory.

Assuming perfect observability, the POMDP formulation collapses to a Markov decision process (MDP) that can be solved using dynamic programming to identify an optimal, static policy vector π that prescribes actions by state [21]. Optimal state transition probability matrix P^{Π} is an $n \times n$ matrix compiled row-wise from P such that $p^{\pi}_{i,j} = p_{\pi(i),i,j}$.

We define a resultant set of state transition probabilities P^* reflecting the impact of partial observability as

$$p^*_{i,j} = \sum_{k=1}^{n} o_{i,k} p_{\Pi(k),i,j}$$

to describe expected state transition probability as the sum of state transition probabilities under any action, weighted by action and observation likelihood under optimal system operation.

This resultant transition probability matrix P^* is a powerful tool in predicting how the system will perform. In particular we lean on the description of the transient state transitions in partition Q^*. In accordance with the background on absorbing Markov chains contained in Chapter 2, Q^* reveals the expected number of state visits before attack as given by

$$N^*_0 = (I_{n-1} - Q^*)^{-1}$$

where I_{n-1} is an identity matrix of size $n - 1$. Because we assume that the proposed system is in s_1 when it is first brought online, the first row of N^*_0 provides the expected number of visits to each state before entering s_n (i.e., before an attack succeeds). The sum of this row is τ^*, the number of total state transitions before attack such that the expected state frequency

visit f_s^* is defined as

$$f_s^* = \frac{\beta(0)N_0^*}{\tau^*}. \tag{6.1}$$

Assuming observations are independent of action selection, the expected observation visit frequency f_ω^* is defined as

$$f_\omega^* = f_s^* O. \tag{6.2}$$

We also define expected attack suppression as

$$\phi^* = 1 - \frac{\tau_1}{\tau^*}. \tag{6.3}$$

Though ϕ^* is not used as a real-time performance correction metric, it is useful in analysis of how well we expect a specific POMDP formulation to meet design objectives.

The expected action visit frequencies f_a^* can also be determined from the optimal policy Π and f_s^*, but the impact of the partial observability is reflected, so the calculation of f_a^* requires a permutation of O in which p_M and p_{FA} are swapped, abbreviated $O^{'}$. This formula was developed from assessment of empirical data from the validation scenario detailed in Chapter 7. The calculation also requires Π, an $m \times n$ matrix representation of the MDP-based policy vector π wherein

$$\Pi_{i,j} = \begin{cases} 1 & \pi(j) == i \\ 0 & \text{otherwise} \end{cases}.$$

A single row of the matrix Π is denoted as Π_m and is used to define the number of expected instances of each action before absorption N_a^* as

$$N_a(m)^* = (\Pi_m O^{'})(\beta(0)N_o^*\zeta)$$

where ζ is a column vector of ones such that f_a^* follows according to

$$f_a(m)^* = \frac{N_a(m)^*}{\sum_{i=1}^{m} N_a(i)^*}. \tag{6.4}$$

The expected overhead per action is defined via the dot product

$$\chi^* = f_a^* \cdot c_{def} \tag{6.5}$$

where c_{def} is the overhead vector described in Chapter 3.

6.2 Setting Thresholds

The error between the model and the real world is identified by assessing the agreement between the expected and the actual values in each of the three metrics, calculated as in Equations 3.1, 3.2, and 3.3. Thresholds for these errors are necessary to identify when POMDP reformulation should occur. The general guidelines for threshold determination are illustrated in the context of the validation scenario thoroughly explored in Chapter 7. The performance monitoring scheme is intended to identify the existence of errors in C, P, and O, though error classification does not occur. Errors in s and ω are not handled by the performance monitoring scheme. Given that a is set by the defenses implemented by the defender, errors in a are not considered possible.

Threshold determination begins by identifying trends in error vector ϵ using simulation of system operation under ideal conditions in which $O = I_n$ and perfect model effectiveness exists. In this case, f_ω, f_a, and χ stabilize near the expected values within approximately 200 decisions, helping to support their utility in quickly identifying model errors. This is equivalent to approximately 30 minutes of system operation assuming a connection inter-arrival time of 10 seconds. This rapid stabilization is displayed in Figures 6.2, 6.3, and 6.4. In each figure, the x-axis represents the number of decisions since initialization while the y-axis represents a single member of set of values averaged to calculate f_ω, f_a, and χ, respectively. The blue lines are the expected value for each member, following Equations 6.2, 6.4, and 6.5, while the black lines are the actual values recorded from the simulation. All three errors approach the expected value following initial fluctuation and align closely by action index 200.

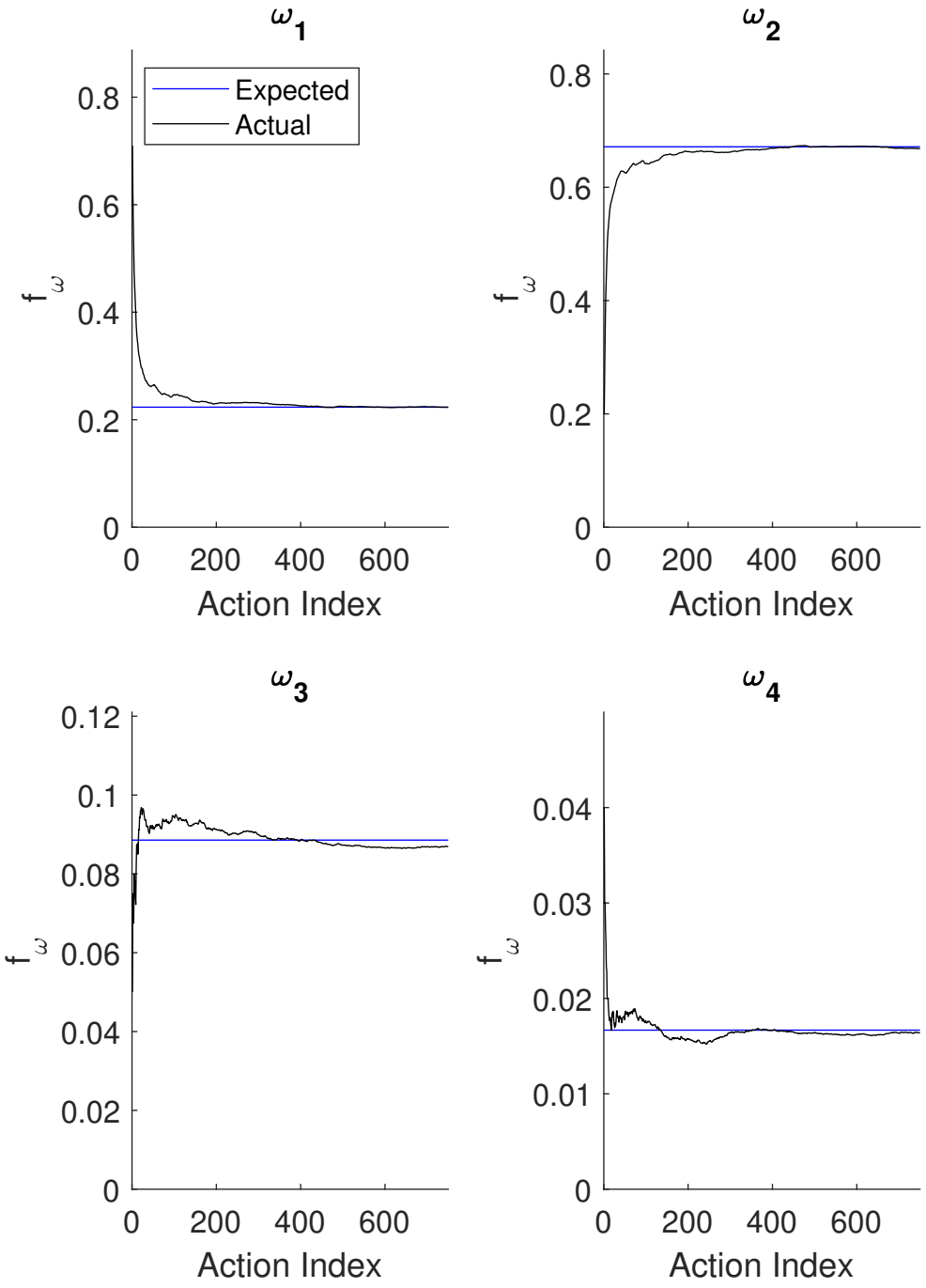

Figure 6.2. Frequency of observation f_ω stabilizes within approximately 200 decisions.

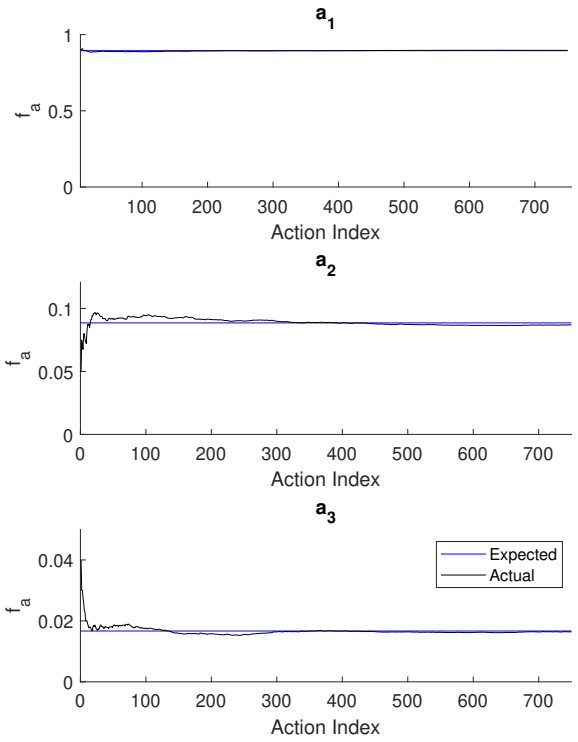

Figure 6.3. Frequency of action f_a stabilizes within approximately 200 decisions.

As might be expected, similar rapid stabilization occurs in all three members of error vector ϵ as displayed in Figure 6.5. Determination of threshold vector δ requires the trade off analysis between false alarm and missed detection inherent in anomaly detection schemes. Depending on the shape of the underlying probability mass function for each variable, mean and standard deviation or median and quantile offer starting places toward tailored threshold determination for a particular application. For illustration, the performance monitoring scheme applies thresholds at the 75th percentile of each metric at the 750th action across 100 simulated runs of the proposed system under perfect model effectiveness.

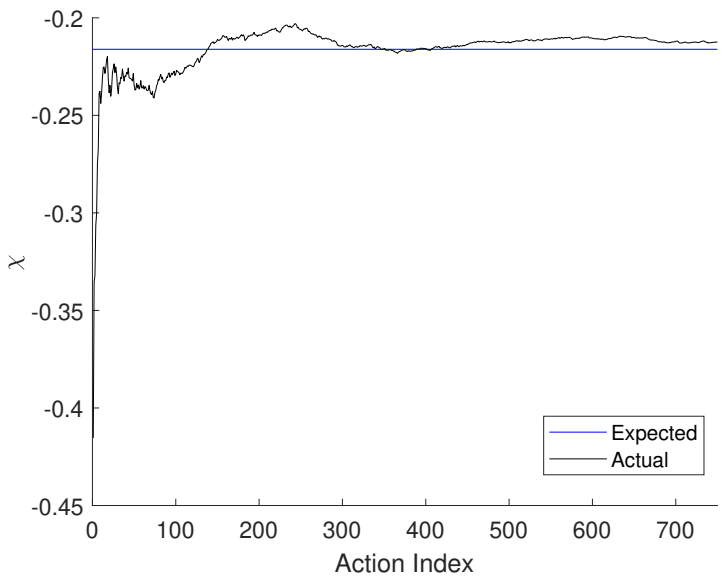

Figure 6.4. Accumulated overhead per action χ stabilizes within approximately 200 decisions.

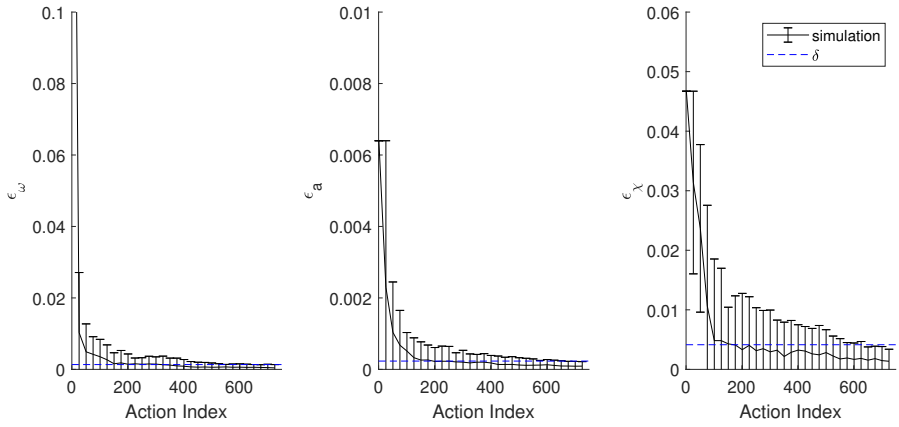

Figure 6.5. Error metrics in ϵ stabilize within the first 200 decisions under ideal conditions. In this example, δ is set at the 75th percentile across 100 ideal trials.

6.3 Detection of Attack

While there may be indications of completed attacks from outside of the proposed system, the performance monitoring scheme will also provide an indication that attack end state s_n has been reached as ϵ will approach ϵ_{atk}, a predictable error vector, once the system enters s_n.

The first element $\epsilon_{atk,\omega}$ of error vector ϵ_{atk} is constrained by the fact that f_ω and ϵ_ω are calculated across the $n-1$ transient states only. As such, once the system enters s_n and the repeated observations of ω_n arrive, $\lim_{t\to\infty} f_{\omega,i}(t) = 0$ for $1 \leq i \leq n-1$. Substituting this expression in Equation 3.1, we define

$$\epsilon_{atk,\omega} = \frac{1}{n}\sum_{i=1}^{n-1} {f^*_{\omega,i}}^2.$$

The action metric ϵ_a also reaches a predictable value. Once in s_n, only a_1 will be viable, as the absorbing state ensures the other actions incur overhead for no gain. Thus, the action visit frequency stabilizes as

$$\lim_{t\to\infty} f_{a,m}(t) = \begin{cases} 1 & m = 1 \\ 0 & m > 1 \end{cases}$$

which when substituted into Equation 3.2 results in the definition of $\epsilon_{atk,a}$ as

$$\epsilon_{atk,a} = \frac{1 - 2f^*_a(1) + \sum_{i=1}^{m} f^*_a(i)^2}{m}.$$

Similarly, $\epsilon_{atk,\chi}$ can be defined based on Equation 3.3 as

$$\epsilon_{atk,\chi} = (\chi^*)^2$$

because $\chi(t) = 0$ once $s(t) = s_n$.

In supporting these definitions, we looked at the agreement between ϵ_{atk} and ϵ in the validation scenario for 30 trials in which the system was fully compromised at $t = 500$ such that $s(t) = s_n$ for $t \geq 500$.

As displayed in Figure 6.6, ϵ approaches ϵ_{atk} gradually over approximately 9,000 decisions. Thus, this is an effective means of detecting attack, though it is not rapid. Under the conditions of the simulation, 9,000 decisions equates to approximately 25 hours of time. Direct use of observations of ω_n is a more rapid indication of having entered s_n.

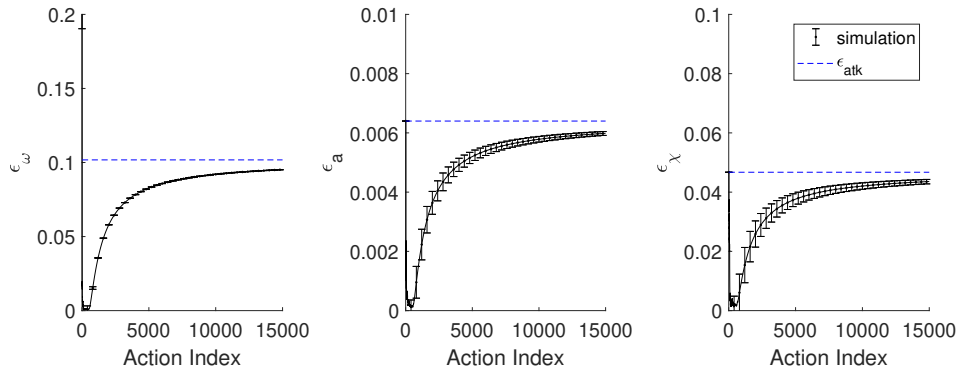

Figure 6.6. The performance monitoring scheme provides indications of attack as $\epsilon \longrightarrow \epsilon_{atk}$. In these trials, $s(t) = s_n$ for $t > 500$.

6.4 Examples of Error Detection

From error vector ϵ and threshold vector δ, the system is capable of detecting errors in model components C, O, and P. This section describes the characteristics of ϵ in each case.

6.4.1 Error in C

Error in C is caused by an incorrect assessment of defensive cost measures and will be apparent if $\epsilon_o < \delta_o$, $\epsilon_a < \delta_a$, but $\epsilon_\chi > \delta_\chi$. An example of the metrics under this error is displayed in Figure 6.7. Because the defenses are completely controlled by the defender, this error is straightforward in that POMDP reformulation requires an update of C with the values observed in system operation.

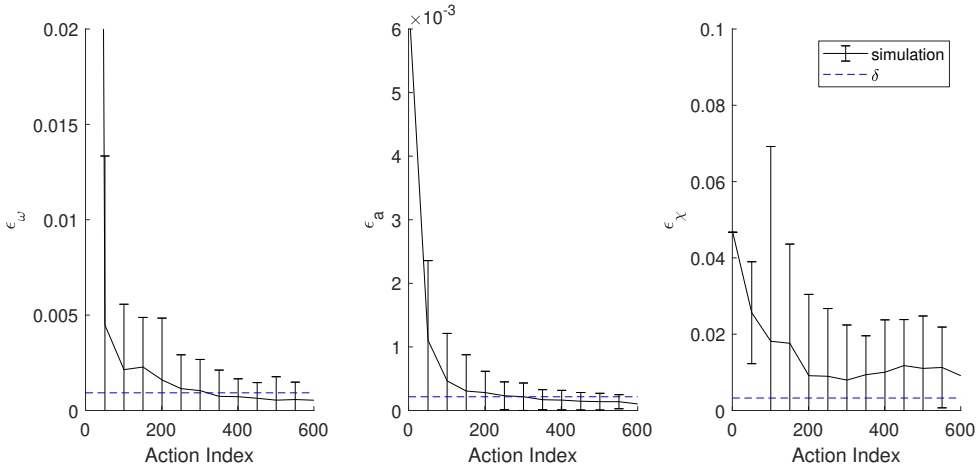

Figure 6.7. The performance monitor detects error in C as $\epsilon_o < \delta_o$, $\epsilon_a < \delta_a$, but $\epsilon_\chi > \delta_\chi$.

6.4.2 Error in O or P

This section looks at errors in IDS or attack analysis that result in errors in O and P. Such errors impact all three components of ϵ. This overlap hinders classification of error source for these cases.

Error in P can be of two types. The first type, error in P_1, stems from an inaccurate understanding of the attacker. To look at the impact of such an error, we imposed an error ranging from $[-0.06, 0.2]$ on $p_{1,i,i}$, the probability that the system remains in a given state at the next transition, with error displaced to $p_{1,i,i+1}$. We then tracked the error metric across 100 trials of 750 decisions each. The median, 25th, and 75th percentiles of each error metric taken at the 750th decision across 100 trials under each error condition are displayed in Figure 6.8. There is a tolerance of around 0.05 in either direction before the thresholds are broken. Thereafter, however, ϵ increases proportional to the error and indicates POMDP reformulation should occur.

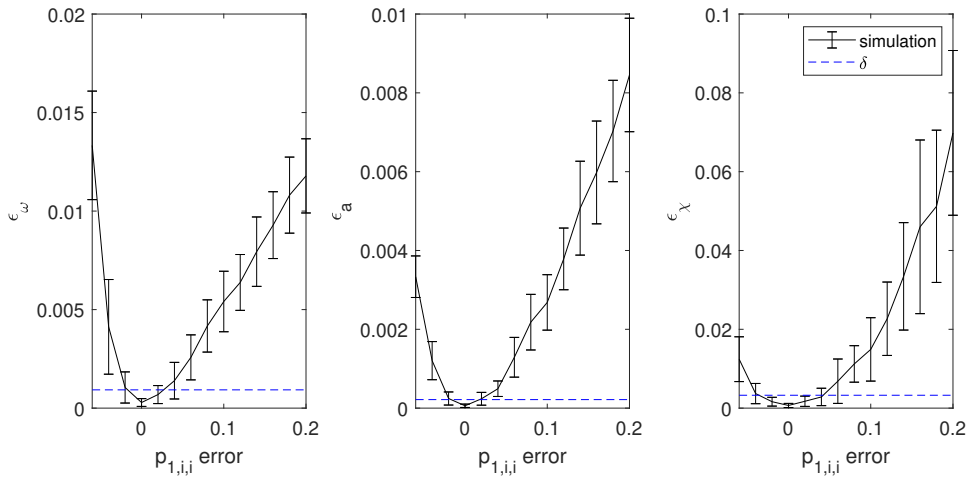

Figure 6.8. Error metrics in ϵ increase proportional to the absolute value of errors in $p_{1,i,i}$.

Error in P_i for $i > 1$ can also stem from an inaccurate understanding of effectiveness of a_i in thwarting attack progress. The timeliness of detecting such errors is related to the frequency with which a_i occurs as it must be selected often enough to influence ϵ. Thus, infrequently used actions should be validated via other means.

Error in O results from incorrect assessment of the upstream IDS. To review the impact of this error, we conducted simulation of the system operating with error in p_D in the range $[0, 0.2]$, again looking at ϵ across 100 trials of 750 decisions each. These results are displayed in Figure 6.9 and are similar to those from the P_1 error investigation, as ϵ increases proportional to the error. Again, the system exhibits an error tolerance of approximately 0.05 before the thresholds are broken.

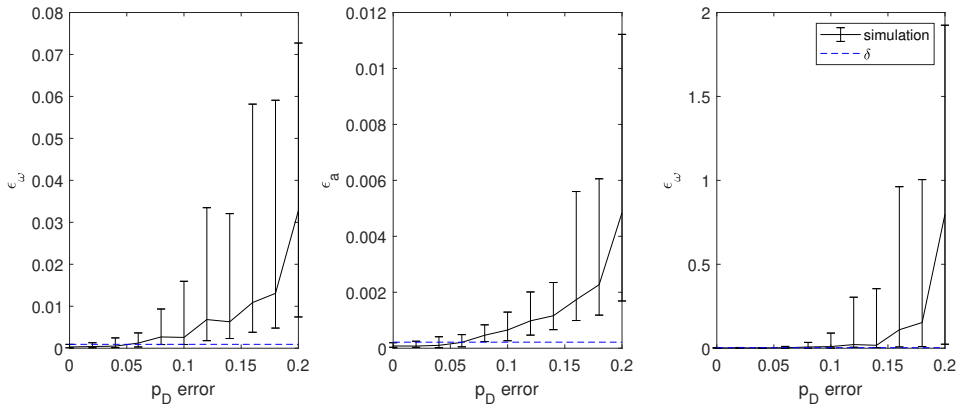

Figure 6.9. Error metrics in ϵ increase proportional to errors in p_D.

6.4.3 Other Model Errors

Errors in s and ω are possible if the phases of attack are different than those determined during attack analysis. These errors are likely to observed via ϵ_ω. Addition of extra members in s and ω would result in those states and observations never being visited or received. Omission of members of either set would result in the system loitering in phases longer than might be expected as some additional intermediate activity occurs that the system is not capable of monitoring.

6.5 Summary

The performance monitoring scheme is unique to our proposed system as compared to the other POMDP-based cyber defense systems in the literature. Identifying a process by which model error can be detected encourages reliance on model-based defensive systems as corrective action can be taken well ahead of catastrophe. This chapter concludes the discussion of our solution approach. The next chapter validates the proposed approach by quantifying system performance in simulation.

CHAPTER 7:
Results

This chapter examines how the proposed system would defend against a five-stage attack by optimally choosing between three actions. We first describe how the proposed system was implemented for validation purposes and present results from simulation of that implementation. Next, we compare our system to a state-of-the-art MTD system in which reconfigurations occur pseudo-randomly with exponentially distributed inter-arrival times. Finally, we explore how the proposed system performs under conditions of model error. Preliminary analysis of the proposed system under this validation context was detailed in [80]; the discussion in this chapter expands on those initial findings.

7.1 System Implementation

The phased attack model studied in validation is illustrated in Figure 7.1. The attack has five phases, i.e., $|s| = 5$ in which states $\{s_1, s_2, s_3, s_4, s_5\}$ correspond to the progression from *start* to *attacked*. This attack model aligns with the five-phase attack described in Chapter 2 with the following adjustments: First, a *start* phase is added to describe the defended system before attack activity begins. Further, we assume that *development* has occurred out of sequence so that the attacker has an exploit packaged and ready to launch when the attack begins and thus leave this phase out of the model. We selected this progression for validation of the proposed system because it was also used to study the effectiveness and overhead of MTD in [8] and thus will permit comparison between our proposed system and the state-of-the-art.

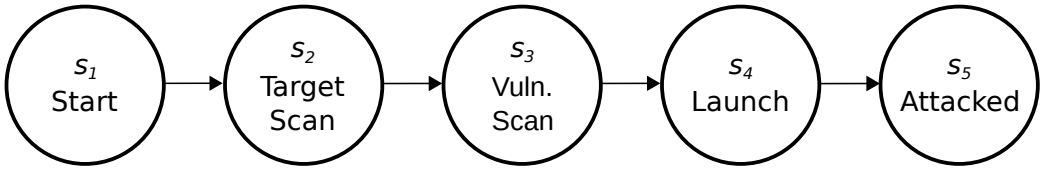

Figure 7.1. Five phase attack process used for validation of proposed system

In *start* s_1 no attacker is yet working against the system. Next, *target scan* s_2 aligns with the

reconnaissance phase described in Chapter 2 and represents the initial efforts performed by the attacker to locate the system. An example of this type of activity is an internet control message protocol (ICMP) *ping* sweep used by an attacker to identify all hosts in range of network addresses. If the system enters this phase, activity has been detected that indicates an attacker has located the system. The next phase is *vulnerability scan* s_3 representing reconnaissance efforts meant to identify particular vulnerabilities of the protected system, such as might be found via the network mapping tool NMAP [42]. This phase aligns with the access phase described in Chapter 2. Finally, in *launch* s_4 the attacker actually attempts a compromise. If successful, the compromise results in the protected system entering the *attacked* state s_5 which aligns with the description of the persistence phase described in Chapter 2.

7.1.1 POMDP Formulation

This section steps through how each of the parameters $\{s, a, \omega, P, C, O, \gamma\}$ were formulated during validation of the proposed system. We used objective analysis of forensic attack data, defense specification, and performance requirements to the maximum extent possible.

Attack Analysis

The attack model illustrated in Figure 7.1 was used to define s, which is the first step of attack analysis. We sought published data from real-world attacks from which P_1 could be formulated. We translated honeypot data into a Markov chain modeling attacker dynamics. The data used are from [17] and were collected over 48-days as attackers interacted with two honeypot web servers behind a university firewall.

The authors of [17] categorized activity as one of four different events based on a count of packets-per-connection as specified in Table 7.1. These classifications were possible because the honeypots served no true function, and thus there was no valid reason to communicate with them. Two protocols were identified in traffic, namely ICMP and transmission control protocol (TCP). All ICMP traffic was assumed to be scanning activity, regardless of packet volume. The TCP traffic was of three types: the lowest volume connections are identified as port scans, intermediate volume as vulnerability scans, and high volume as launches. The specific thresholds between categories were established by studying the traffic pattern of well-known methods of performing each function.

Table 7.1. Attack stage by packets-per-connection, from [17].

Event	Packets per connection	Protocol
ICMP Scan	N/A	ICMP
Port Scan	< 5	TCP
Vuln. Scan	$5 - 12$	TCP
Launch	> 12	TCP

Over the 48 days, 59,468 connections were collected, 22,710 of which went to the two honeypots. Of those honeypot connections, 6,203 unique records occurred, representing 5,540 individual attacks. Each attack progression occurred with the counts listed in Table 7.2. A four digit binary value represents the attack progression that from left to right signifies *launch – vulnerability scan – port scan – ICMP scan*. For example, 1000 represents an occurrence of s_4 with no indication of earlier scan activity. Although the authors do not distinguish order of events in cases with more than one type of scan, we assume that scans always occur in the order *ICMP scan – port scan – vulnerability scan* given that this corresponds with an increasing level of detail afforded to the attacker.

Table 7.2. Occurrences of each attack progressions in which each progression is comprised of a series of connections. Adapted from: [17].

Progression	No. Observed	Progression	No. Observed
0000	10,807	1000	381
0001	2,796	1001	1
0010	666	1010	28
0011	11	1011	0
0100	1,103	1100	296
0101	179	1101	5
0110	17	1110	42
0111	8	1111	7

Note: Big endian binary values are used to label attack progressions such that from left to right, values correspond to *launch - vulnerability scan - port scan - ICMP scan* with 1 indicating occurrence and 0 the opposite.

To align the available data in [17] with our five state attack model, we combined the two low fidelity scans into the *target scan* state. Thus, we combine the values of Table 7.2 into those displayed in Table 7.3. Now, a three digit binary value represents the attack progression that from left to right signifies *launch - vulnerability scan - target scan*. For example, 100 represents detection of a launch with no indication of reconnaissance scans having been performed.

Table 7.3. Occurrences of attack progression collapsed to the attack model in Figure 7.1. Adapted from: [17].

Progression	No. Observed	Progression	No. Observed
000	10,807	100	381
001	3,473	101	29
010	1,103	110	296
011	204	111	54

Note: Big endian binary values are used to label attack progressions such that from left to right, values correspond to *launch - vulnerability scan - target scan* with 1 indicating occurrence and 0 the opposite.

Under the assumption that the attack stages may be skipped but may not occur out of order from s_1 through s_5, these observation counts translate into the Markov chain in Figure 7.2 with values from Table 7.4. In calculating these probabilities, we assume that attacks occur via consecutive connections without intermediate traffic from other sources. We apply a per-connection basis for state transitions.

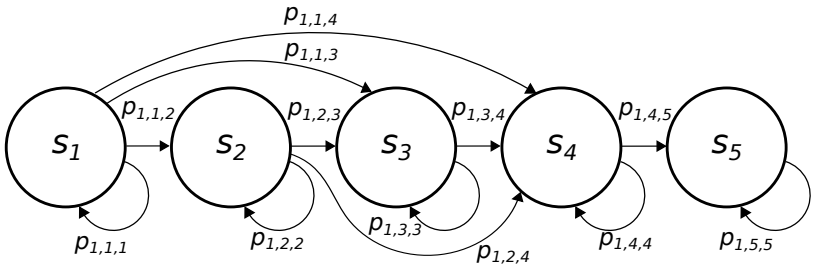

Figure 7.2. Markov chain of cyber attack process used in validation.

We extended the model to the fifth state to account for the failure of exploits to take effect and cause damage, which was not measured in [17]. In the presence of layered defense, $p_{1,4,5}$ represents the probability of all other defenses (e.g., anti-virus software, privilege control, security training) failing. For validation, we applied a value of 0.5 to conservatively represent the likelihood another layer of defense prevents the launch from being successful.

Table 7.4. Mapping of occurrences observed in [17] into transition probabilities for a five state Markov chain

s	arrival count	departure to s_1		s_2		s_3		s_4		s_5	
		count	$p_{1,i,1}$	count	$p_{1,i,2}$	count	$p_{1,i,3}$	count	$p_{1,i,4}$	count	$p_{1,i,5}$
s_1	16,347	10,807	0.661	3,760	0.23	1,399	0.086	381	0.0230	0	0.0
s_2	3,760	0	0.0	3,473	0.924	258	0.069	29	0.008	0	0.0
s_3	1,657	0	0.0	0	0.0	1,307	0.789	350	0.211	0	0.0
s_4	760	0	0.0	0	0.0	0	0.0	380	0.5	380	0.5
s_5	380	0	0.0	0	0.0	0	0.0	0	0.0	380	1.0

Together, the probabilities form state transition probability matrix

$$P_1 = \begin{bmatrix} 0.6611 & 0.2300 & 0.0856 & 0.0233 & 0 \\ 0 & 0.9235 & 0.0687 & 0.0078 & 0 \\ 0 & 0 & 0.7900 & 0.2100 & 0 \\ 0 & 0 & 0 & 0.5000 & 0.5000 \\ 0 & 0 & 0 & 0 & 1.0000 \end{bmatrix}. \tag{7.1}$$

As the first member of P, P_1 captures a number of qualities of the underlying attack process. The standard form of P_1 follows Equation 2.5 for one absorbing state and $n-1=4$ transient states so that Q_1 is the upper left partition of the state transition probability matrix comprising only the transient states. Given the specific values in Equation 7.1, the fundamental matrix

N follows Equation 2.6 such that

$$N = \left(I_4 - \begin{bmatrix} 0.6611 & 0.2300 & 0.0856 & 0.0233 \\ 0 & 0.9235 & 0.0687 & 0.0078 \\ 0 & 0 & 0.7900 & 0.2100 \\ 0 & 0 & 0 & 0.5000 \end{bmatrix} \right)^{-1}$$
$$= \begin{bmatrix} 2.951 & 8.872 & 4.105 & 2 \\ 0 & 13.072 & 4.276 & 2 \\ 0 & 0 & 4.762 & 2 \\ 0 & 0 & 0 & 2 \end{bmatrix}.$$

We assume that the system begins in s_1, i.e., $b(0) = [1, 0, 0, 0, 0]$, and thus follow Equation 2.7 to determine that there are an expected $\tau_1 = 17.928$ total state visits before reaching the *attacked* state. This is a critical metric as it serves as a baseline for quantifying the improvement in attack suppression of any defensive system installed. Moving into the simulated operation of the system, we assume that connections arrive every 10 seconds such that decisions occur at the same rate. Thus, absent defense, the system would be attacked within approximately 3 minutes.

From Equation 2.8, the variance N_2 for attack model P_1 is

$$N_2 = \begin{bmatrix} 5.7560 & 144.3593 & 18.1392 & 2.0000 \\ 0 & 157.8026 & 18.1636 & 2.0000 \\ 0 & 0 & 17.9138 & 2.0000 \\ 0 & 0 & 0 & 2.0000 \end{bmatrix}.$$

Comparing $N_{sq} = N \circ N$ and N_2 provides an indication of the reliability of the means as estimates for a given Markov chain [20]. For the system in question,

$$N_{sq} = \begin{bmatrix} 8.7084 & 78.7124 & 16.8510 & 4.0000 \\ 0 & 170.8770 & 18.2842 & 4.0000 \\ 0 & 0 & 22.6766 & 4.0000 \\ 0 & 0 & 0 & 4.0000 \end{bmatrix}.$$

These values are in general agreement with N_2 in that all values are of the same order of magnitude. Thus, N offers reliable estimates of expected performance. However, τ_1 and N are the expectation of variables with long right tailed distributions, so individual samples can vary significantly from the expected value. The implications are that median will be more useful than mean in establishing thresholds for the performance monitoring scheme.

To explore this practically for the attack model used in validation, we conducted 100,000 simulated attacks, compiling the spread of state visits across all simulations. The significant right tail is visible in the values in Table 7.5 and the histograms in Figures 7.3 and 7.4.

Table 7.5. Volume of state visits before absorption into s_5 with theoretical values following Equation 7.1.1.

State	theoretical value		Across 100,000 Simulated Attacks, $p_D = 1.0$						
						(a)		(b)	
	$E[visits]$	$Var[visits]$	mean	var.	median	max	min	max	min
s_1	2.95	5.76	2.96	5.81	2	28	1	2	1
s_2	8.87	144.36	8.85	143.26	4	144	0	144	0
s_3	4.11	18.14	4.10	18.34	3	49	0	4	0
s_4	2.00	2.00	2.01	2.02	2	15	1	1	2
total	17.93	170.25	17.92	173.13	14	236	2	151	3

Note: The max/min values in column (a) are across all data, while those in column (b) are drawn from a single attack progression.

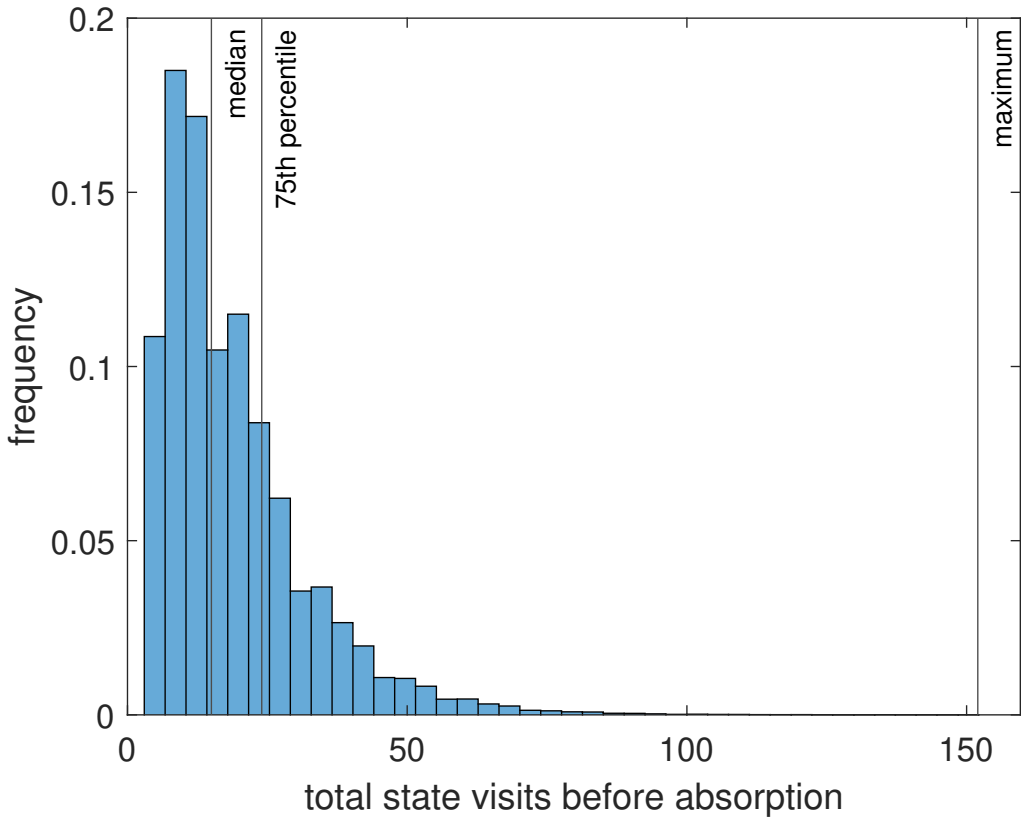

Figure 7.3. Occurrence of total transient state visits before absorption over 100,000 simulated attacks, $p_D = 1.0$.

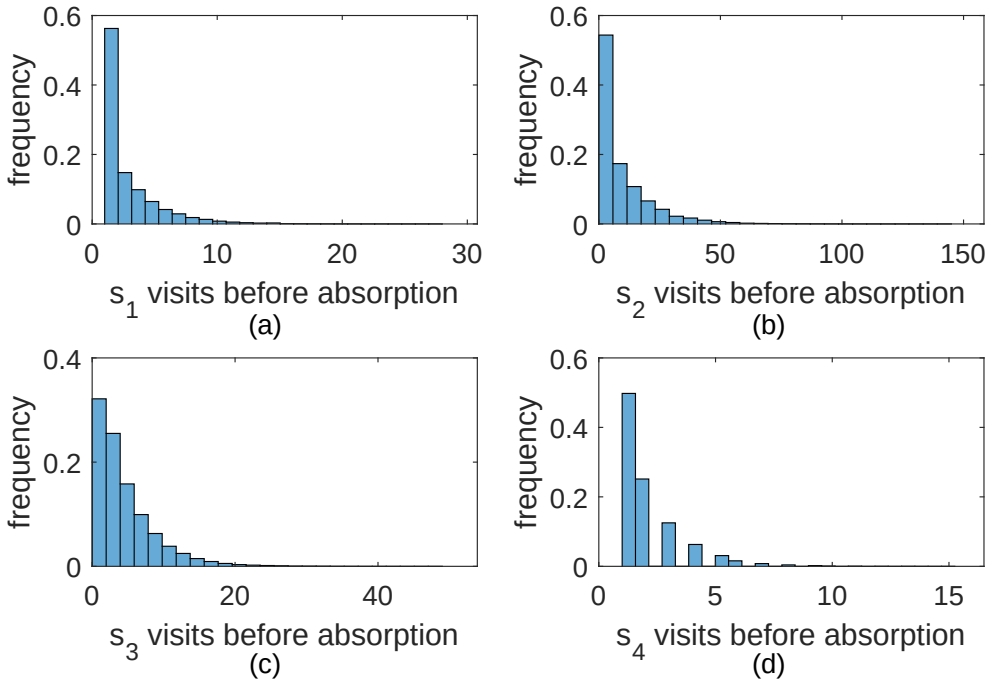

Figure 7.4. Occurrence of individual state visits before absorption over 100,000 simulated attacks, $p_D = 1.0$.

Defense Analysis

For the validation scenario, we implemented three possible MTD, i.e., $|a| = 3$. While there are many more defenses available, both from the catalog of MTD techniques detailed in [6] and other options like restarting resources or partitioning the network connectivity of devices under attack, implementing these three defenses for validation facilitated a direct comparison between our system and the state-of-the-art system detailed in [8].

The first, a_1, represents a *nil* defense in which the system continues to follow attacker whim as described by P_1. The remaining defenses are modeled after real-world MTD strategies with details drawn from [8]. When effective, each returns the system to s_1. Following the generic pattern depicted in Figure 3.1, a_2 represents a dynamic platform change between μ

different services with repeat such that

$$P_2 = \begin{bmatrix} \frac{\mu-\mu_v}{\mu} & 0 & 0 & 0 & 0 \\ \frac{\mu-\mu_v}{\mu} & 0 & 0 & 0 & 0 \\ \frac{\mu-\mu_v}{\mu} & 0 & 0 & 0 & 0 \\ \frac{\mu-\mu_v}{\mu} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{\mu-\mu_v}{\mu} \end{bmatrix} + \frac{\mu_v P_1}{\mu}$$

wherein μ_v are assumed to be vulnerable to attack.

Similarly, a_3 represents a dynamic network change in IP address between ρ available addresses, with repeat such that

$$P_3 = \begin{bmatrix} \frac{\rho-1}{\rho} & 0 & 0 & 0 & 0 \\ \frac{\rho-1}{\rho} & 0 & 0 & 0 & 0 \\ \frac{\rho-1}{\rho} & 0 & 0 & 0 & 0 \\ \frac{\rho-1}{\rho} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{\rho-1}{\rho} \end{bmatrix} + \frac{P_1}{\rho}.$$

To align with the defenses implemented and measured in the laboratory in [8], we implement a_1, a_2, and a_3 into our model with specifications as recorded in Table 7.6. We measure overhead in terms of availability such that c_i represents the loss in system availability in seconds when MTD a_i is deployed based on experimental values from [8] assuming a 10 second inter-arrival rate between connections.The service reconfiguration a_2 is a rotation between $\mu = 3$ services, wherein $\mu_v = 1$ is vulnerable to attack and imposes a $c_2 = 0.635$ second loss in availability. The IP address reconfiguration reassigns addresses from a pool of 256 options and imposes a $c_3 = 9.590$ second loss in availability.

Table 7.6. Available defenses. Adapted from: [8].

a_i	Reconfiguration Basis	Effectiveness (P[$s' = s_1$])	Effectiveness (%)	Availability Loss (sec)	Availability (%)
a_1	No Action	n/a	n/a	0.000	100%
a_2	Service	$\frac{\mu-\mu_v}{\mu} = \frac{2}{3}$	66.7%	0.635	93.7%
a_3	IP address	$\frac{\rho-1}{\rho} = \frac{255}{256}$	99.6%	9.590	4.1%

The dynamics under each defense described by P_2 and P_3 are absorbing Markov chains that provide insight into the effectiveness of each defense. Following the same flow of analysis used in Equation 7.1.1, a_2 would extend the expected time before attack 15 times over to $\tau_2 = 105$. Certainly a_2 is useful, though a_3 extends the expected time before attack even further to $\tau_3 = 5.25 \times 10^6$, making it by far the more effective defense but also the most expensive. System availability under such consistent use of a_3 would be just 4.1%. The other options either thwart the attack moderately well for moderate expense (a_2) or not at all but for free (a_1). With these values quantified, the POMDP formulation process pivots to establishing system priorities.

Prioritization of Competing Requirements

Recalling that attack penalty is set relative to the most expensive defense in accordance with Equation 4.1, the next step in formulating the POMDP involves selecting the attack penalty scaling factor ν to reflect the defensive priorities. For the context thus described, the locations of the policy shifts as ν increases are shown by state in Figure 7.5. The optimal policy vector π_ν describes the action that should be taken in each state for a given value of ν as determined for the equivalent MDP via policy iteration as implemented in [81]. For approximately $\nu < 100$, $\pi_\nu = [a_1, a_1, a_2, a_3, a_1]$ while for $\nu > 10^4$, $\pi_\nu = [a_3, a_3, a_3, a_3, a_3]$.

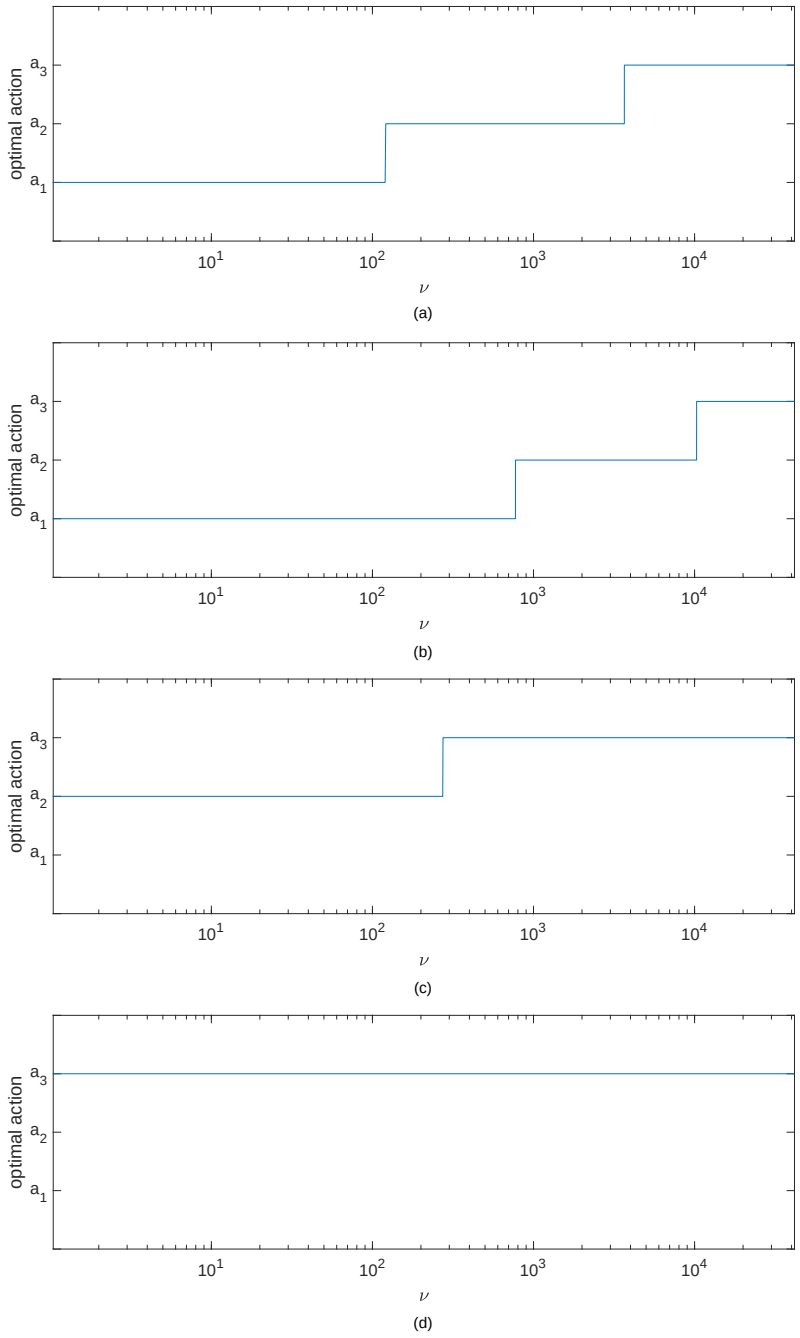

Figure 7.5. Policy shifts as attack penalty scaling factor ν increases. Each plot represents the optimal action in π in a particular state, from (a) s_1 to (d) s_4. The first shift occurs near $\nu = 100$ in (see (a)). By $\nu = 10,000$ the optimization effort is effectively abandoned, as the policy indicates taking the most expensive defense, a_3, in every state.

These two cases represent the policies at either extreme with the former prioritizing overhead minimization and the latter prioritizing attack suppression. This trade-off is illustrated in Figure 7.6 wherein the predicted metrics of attack suppression and availability as a function of attack penalty scaling factor ν are displayed. The stair-step shifts in value align with the policy shifts by state in Figure 7.5. Even for $p_D = 0.5$, it is possible to achieve attack suppression of greater than 99%, but not unless the user is willing to accept availability on the order of 2%. Because the objective of our work is to implement MTD with overhead minimization, we set $\nu = 100$ to explore system performance in the range in which both attack suppression and availability are above 90%. The optimal policy in this case is $\pi_{100} = [a_1, a_1, a_2, a_3, a_1]$, which escalates defensive effort as attack phase progresses in an echo of our original moth-based inspiration.

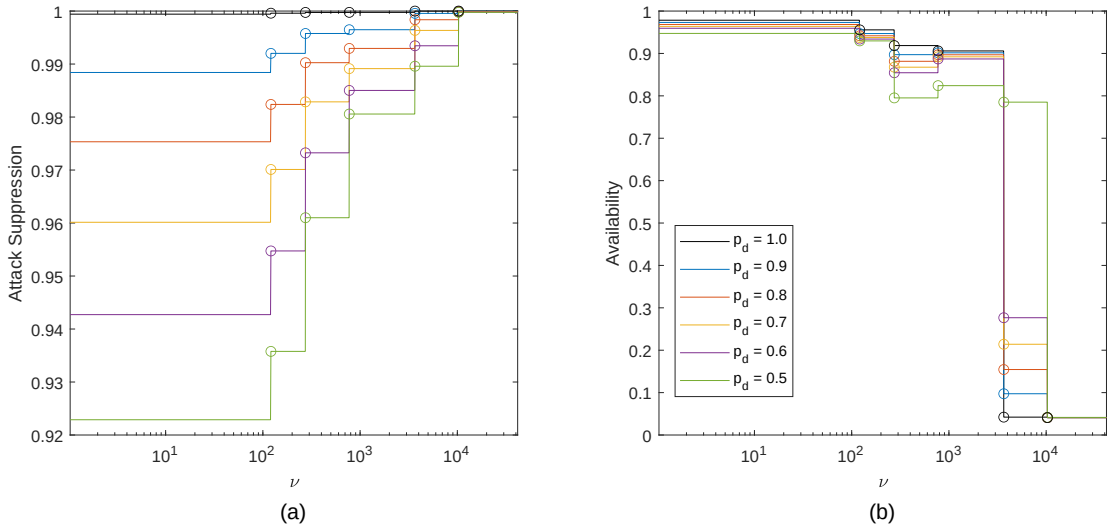

Figure 7.6. Sensitivity of performance metrics (a) attack suppression and (b) availability to variation in attack penalty scaling factor ν.

The discount factor γ influences the priority of immediate versus future rewards. The optimal policy for the system studied in validation is not particularly sensitive to γ as illustrated in the sensitivity analysis displayed in Figure 7.7, which was developed by calculating the optimal policy under each γ via policy iteration as implemented in [81]. At $\gamma = 0$, the optimal policy is $\pi = [a_1, a_1, a_1, a_2, a_1]$ while at $\gamma = 1.0$, $\pi = [a_1, a_1, a_2, a_3, a_1]$. We selected $\gamma = 0.75$ to ensure both long-term attack suppression and near-term availability

were achieved but could have selected any value in the range $0.34 \leq \gamma \leq 0.99$ and achieved similar balance.

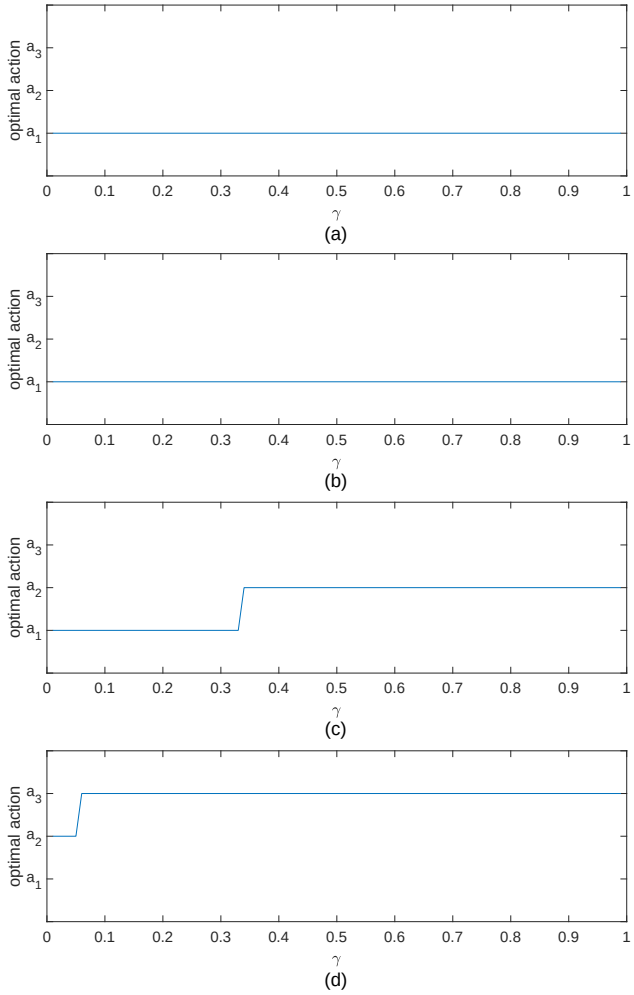

Figure 7.7. Policy shifts across discount factor γ. Each plot represents the optimal action in π in a particular state, from (a) s_1 to (d) s_4. Policy vector π is stable for $\gamma > 0.34$ as indicated by static action-state pairing for $0.34 < \gamma \leq 1.0$.

Partial Observability

Because the intrusion detection system is upstream of the proposed system, our work explores the performance of the proposed system across a range of capabilities for that upstream system. We consider intrusion detection system performance to be independent of the proposed system such that $O_i = O_j$ for $i, j \in [1, ..., m]$. We set $\omega \equiv s$ and explore three uncertainty conditions: U_1, U_2, and U_3, all of the form of Equation 4.2 over a range of probability of detection p_D values. In this system, p_D measures the likelihood that $s(t) = \omega(t)$.

The first case, U_1, lumps all IDS error in false positive detection of the next highest numbered state such that

$$O_{U_1} = \begin{bmatrix} p_D & 1-p_D & 0 & 0 & 0 \\ 0 & p_D & 1-p_D & 0 & 0 \\ 0 & 0 & p_D & 1-p_D & 0 \\ 0 & 0 & 0 & p_D & 1-p_D \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The next case, U_2 considers only missed detection such that

$$O_{U_2} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1-p_D & p_D & 0 & 0 & 0 \\ 0 & 1-p_D & p_D & 0 & 0 \\ 0 & 0 & 1-p_D & p_D & 0 \\ 0 & 0 & 0 & 1-p_D & p_D \end{bmatrix}.$$

The final mixed case U_3 considers missed and false detection equally likely such that

$$O_{U_3} = \begin{bmatrix} p_D & 1-p_D & 0 & 0 & 0 \\ \frac{1-p_D}{2} & p_D & \frac{1-p_D}{2} & 0 & 0 \\ 0 & \frac{1-p_D}{2} & p_D & \frac{1-p_D}{2} & 0 \\ 0 & 0 & \frac{1-p_D}{2} & p_D & \frac{1-p_D}{2} \\ 0 & 0 & 0 & 1-p_D & p_D \end{bmatrix}.$$

Note that our attack analysis assumed that $p_D = 1.0$ for the results in [17] such that the

reported attack progressions define P_1, not O. While it would be possible to consider a skipped attack phase as a missed detection, rather than a true indication that the attacker was able to make progress without an intermediate step, we consider our assumption valid because the data were collected from honeypots specifically tailored for attack-phase detection and categorization [17].

This section has outlined POMDP formulation and the values of s, a, ω, P, C, O and γ that are used in the remaining sections of Chapter 7 to explore various aspects of the proposed system in action. For succinct reference, this POMDP formulation is summarized in Appendix A.

7.1.2 MTD Selection

In this subsection, we describe the way in which we configured the MTD selection subsystem so that it effectively operated as POMDP agent. DESPOT parameters were set at $\{d, q, z\} = \{11, 80, 40\}$. The first parameter, depth d, was set at 11 based on analysis of τ_1 as described in Section 5.1. For all simulations, we set $z = 40$ so that the actual clock time DESPOT used to perform simulations toward selecting the optimal MTD was limited to 40 seconds. It was necessary to select a value high enough to permit deep exploration without becoming unreasonable in terms of the memory available on the simulation hardware. The specifications of the hardware used in validation are provided in Appendix C.

The final parameter, particle count q, was selected by conducting exploratory simulations of 100 decisions each across a range of values to identify that which would result in optimal decisions in a majority of cases. The results of these exploratory trials are presented in Table 7.7. At $p_D = 1.0$, the perfect optimal decision is explicitly known as the POMDP collapses to an MDP that can be solved for an optimal policy. Thus, we calculate a probability of achieving the optimal decision by comparing 100 DESPOT decisions at each test point to what was prescribed by the MDP policy vector $\pi = [a_1, a_1, a_2, a_3, a_1]$.

Table 7.7. Impact of *particles* parameter on probability of achieving optimal decision via DESPOT, *depth* = 11, *time* = 40, $p_D = 1.0$.

state	particle count					
	20	40	80	160	320	640
s_2	1.00	1.00	1.00	0.99	0.89	0.91
s_3	0.79	0.92	0.98	0.99	0.73	0.77
s_4	0.99	1.00	1.00	1.00	1.00	1.00

The drop off in accuracy with increase in particle count is a result of the inability of the system to explore all particles out to the designated d for a given z. To stay well back from this limitation, system trials were conducted at $q = 80$.

The final parameter test verified that the selected values $\{11, 80, 40\}$ would sustain quality decision making as p_D degraded. The results of these trials are summarized in Table 7.8. In these trials, the system began in the indicated state with probability of 1.0, meaning that partial observability impacted simulations conducted to explore future actions and observations but not the starting belief state. As indicated, the selected parameters sustained high-quality decision making through $p_D = 0.8$.

Table 7.8. Impact of p_D on probability of achieving optimal decision via DESPOT, $\{d, q, z\} = \{11, 80, 40\}$

state	p_D						
	1.000	0.975	0.950	0.925	0.900	0.850	0.800
s_2	1.00	1.00	1.00	1.00	0.99	1.00	1.00
s_3	0.98	1.00	0.99	0.97	0.97	0.97	0.96
s_4	1.00	1.00	1.00	1.00	1.00	1.00	1.00

7.1.3 Results

With the POMDP model formulated, the next step in validation was to ensure the system could optimally defend against attack as designed. To test this aspect, we operated the system

across three uncertainty models as described in Section 7.1.1. In each case and at each p_D, data were collected across a decision volume $\eta_{dec} = 330,000$ decision cycles, which was equivalent to attack volume η_{atk} in the range $36 \leq \eta_{atk} \leq 235$ depending on the success of the system in thwarting the attacker. Specific trial lengths per data point are recorded in Table 7.9.

Table 7.9. Attack volume η_{atk} and decision volume in millions η_{dec} used to quantify simulated system performance.

	p_D											
	1.0		0.98		0.92		0.86		0.80		0.74	
case	η_{atk}	η_{dec}	η_{atk}	η_{dec}	η_{atk}	η_{dec}	η_{atk}	η_{dec}	η_{atk}	η_{dec}	η_{atk}	η_{dec}
U_1	100	2.77	58	1.29	40	0.88	68	1.34	69	1.24	90	0.90
U_2	100	2.77	36	0.33	91	0.38	132	0.44	137	0.39	195	0.47
U_3	100	2.77	56	0.98	93	0.85	135	0.98	184	0.96	235	0.80

Because it offers a mix of missed and false detection, we use U_3 to describe the results in detail from this point forward with additional results from Cases U_1 and U_2 contained in Appendix B. These simulations were conducted under conditions of ideal model effectiveness in that the POMDP formulation detailed in Appendix A was used within the proposed system and the simulation controller except where noted to explore the implications of model error.

The proposed system maintained mean availability at above 94% and mean attack suppression above 99% for $0.76 \leq p_D \leq 1.00$, as depicted in Figures 7.8(a) and 7.8(b) respectively. For perspective, in the absence of the proposed system, assessment of P_1 indicates the system would be successfully attacked every three minutes. The proposed system prevents enough inbound attacks to increase this rate to every nine and a half hours, even at $p_D = 0.76$.

Two y-axis scales are included so that Figures 7.8(a)–(b) provide visual reference of performance across the range of possible values while Figures 7.8(c)–(d) highlight the difference between measured metrics and those expected based on Equations 6.3 and 6.5. Although measured and expected results align exactly at $p_D = 1.0$. As p_D decreased, agreement decreased proportionately. As depicted in Figure 7.8(c)–(d), the system out-performed ex-

pected attack suppression and under-performed expected availability by a maximum of about three percentage points in either case.

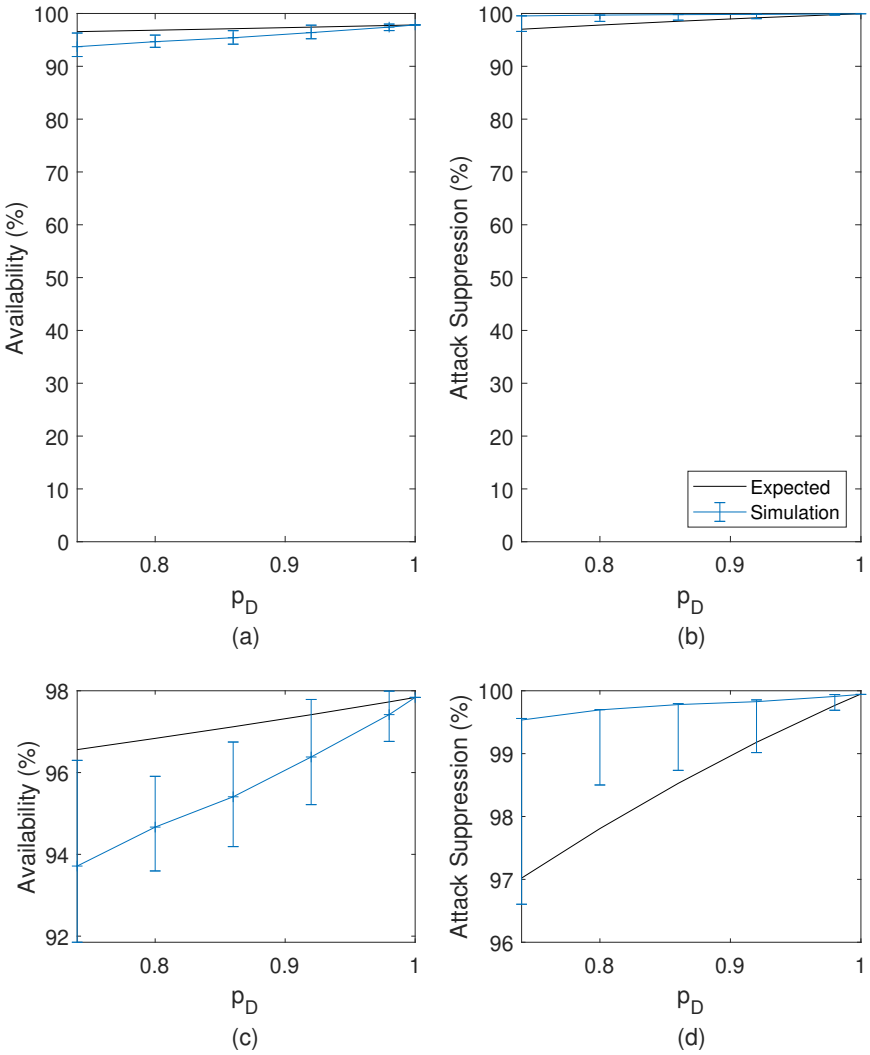

Figure 7.8. Availability and attack suppression performance under the proposed system as p_D degrades for uncertainty case U_3 in which detection error is split between false alarms and missed detection, i.e., $p_{FA} + p_M = 1 - p_D$. The solid black line indicates predicted performance based on Equations 6.3 and 6.5, while the blue line represents the mean across simulated operation of the proposed system, with the 0.25 and 0.75 quantile indicated by the error bars.

These discrepancies between the expected and actual performance metrics are the result of

sub-optimal action selection by the proposed system. As such, the error is also reflected in the trends in action selection presented in Figure 7.9. We repeat the use of two y-axis scales so that Figures 7.9(a)–(c) display results within the range of possible values, while Figures 7.9(d)–(f) expand the region of interest to display the difference between measured and expected values. Expected action visit frequency f_a^* is calculated via Equation 6.4. The tight agreement between expected and simulation results in Figure 7.9(c) indicates a_1 is used optimally even as p_D degrades. Sub-optimal decision making is instead resulting in implementing a_3 when a_2 would be optimal, as the simulation results for a_2 displayed in Figure 7.9(e) are consistently lower than expected by the same amount that the results in a_3 displayed in Figure 7.9(f) are consistently higher than expected. The proposed system chooses a_3, the most effective and most expensive defense, more than expected as p_D degrades which accounts for the measured improvement in attack suppression and degradation in availability.

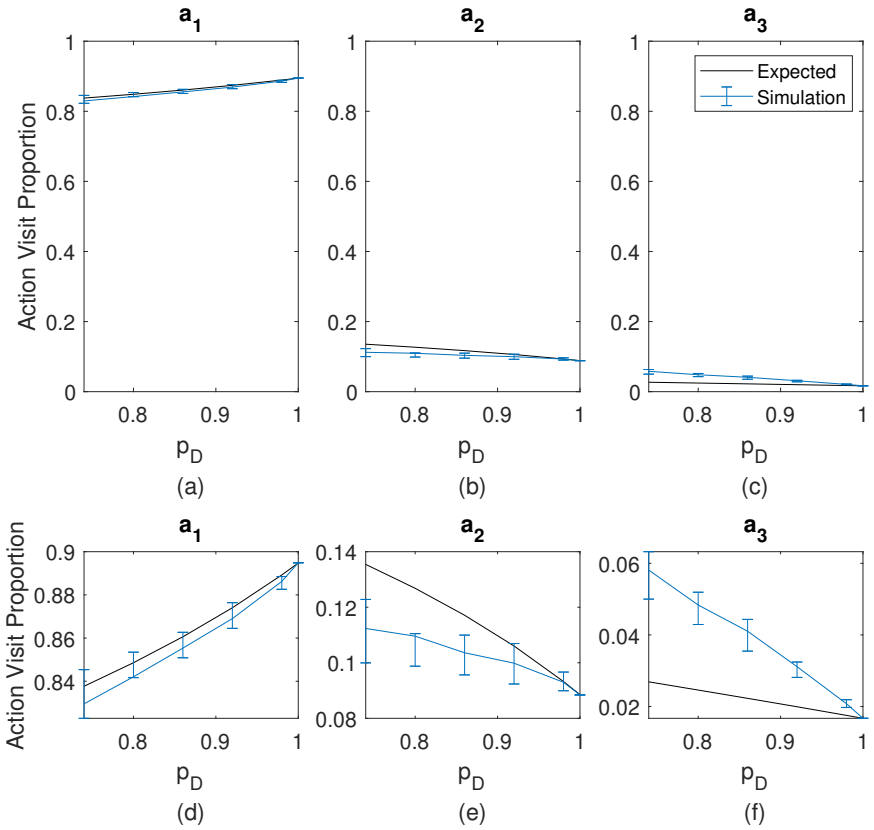

Figure 7.9. Action visit trends under proposed system as p_D degrades for uncertainty case U_3. The solid black line indicates predicted performance based on Equation 6.4, while the blue line represents the mean across simulated operation of the proposed system, with the 0.25 and 0.75 quantile indicated by the error bars.

Further explaining this sub-optimal decision making requires analysis of q_π, the probability that action $a(t) = \pi(s(t))$, the action indicated by the optimal MDP policy vector in true state $s(t)$). Simulation-based measurements of q_π are displayed in Figure 7.10. The dashed line represents $q_\pi = p_D$ in which case there is no advantage to the POMDP-based system over an MDP-based system operated under the assumption that the intrusion detection system is perfect. Overall, as indicated in Figure 7.10, the proposed system outperforms this benchmark, which validates its utility and design overall but does not explain the discrepancy between expected and measured performance metrics.

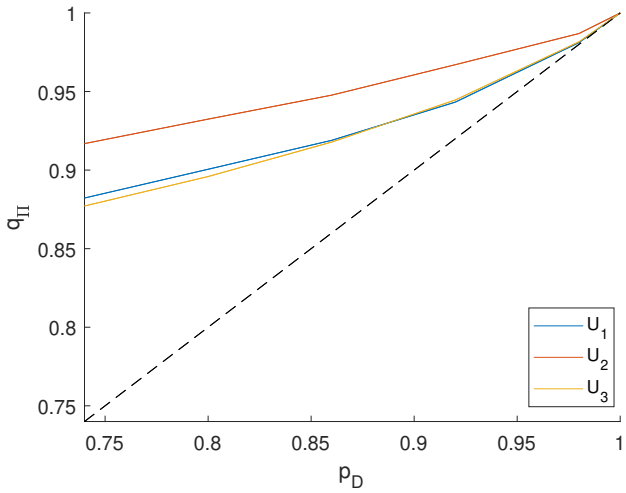

Figure 7.10. The impact of probability of detection p_D on the probability $a(t)$ aligns with the optimal MDP action q_π. The dashed line represents $q_\pi = p_D$.

Fine-grained analysis does explain the discrepancy between metrics. Specifically, values of q_π conditioned on state as plotted in Figure 7.11 indicate that the problematic state is s_3. In s_1, s_2, and s_4, the proposed system does better than the benchmark as displayed in Figures 7.11(a),(b), and (d); however, in s_3 far more sub-optimal decisions occur as q_π is up to 0.25 below the benchmark as displayed in Figure 7.11(c).

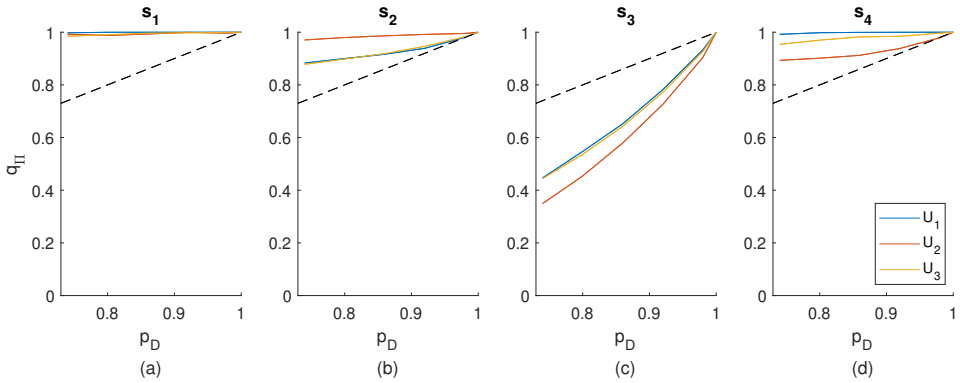

Figure 7.11. The impact of probability of detection p_D on the probability $a(t)$ aligns with the optimal MDP action q_π conditioned on state s_i. The dashed lines represent $q_\pi = p_D$.

The issue in s_3 is further explained by the projected value of decisions a_2 and a_3 in the belief space around s_3. Despite a_2 being the optimal action to take in s_3, a_3 becomes the optimal decision based on belief state for anything beyond s_3. To illustrate, we developed a vector-based policy for $p_D = 0.8$ via the incremental pruning algorithm [73] as implemented in *pomdp-solve* [82]. This policy is depicted in Figure 7.12. Value-vector policies are used by finding the current belief state along the horizontal axis and taking the action that corresponds to the vector of maximum value in that location. As an example, at $b(t) = [0, 0, 0, 1, 0]$, indicating s_4, the red line is maximum and thus a_3 is the optimal decision.

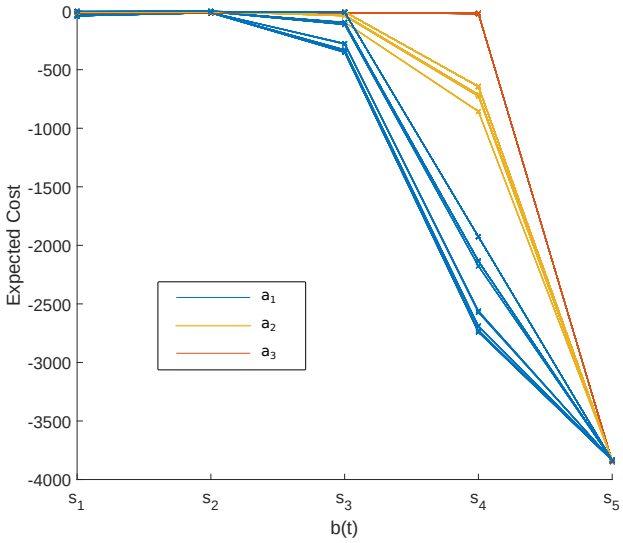

Figure 7.12. Value vector representation of the optimal policy for U_1, $p_D = 0.8$ as developed by applying the incremental pruning algorithm.

The vector-based policy is consistent in the belief state regions around s_1, s_2, and s_4, which facilitates optimal decision making in these belief states even as p_D degrades. A close up of the region immediately around s_3 is displayed in Figure 7.13. While the optimal action when $b(t) = [0, 0, 1, 0, 0]$ is a_2 as indicated by the yellow line, when the belief state $b(t) = [0, 0, 0.975, 0.025, 0]$ the optimal action is a_3 as indicated by the red line. This means that optimal decisions when $s(t) = s_3$ require a belief state within just a few hundredths of the true state, which is only possible when $p_D \approx 1.0$.

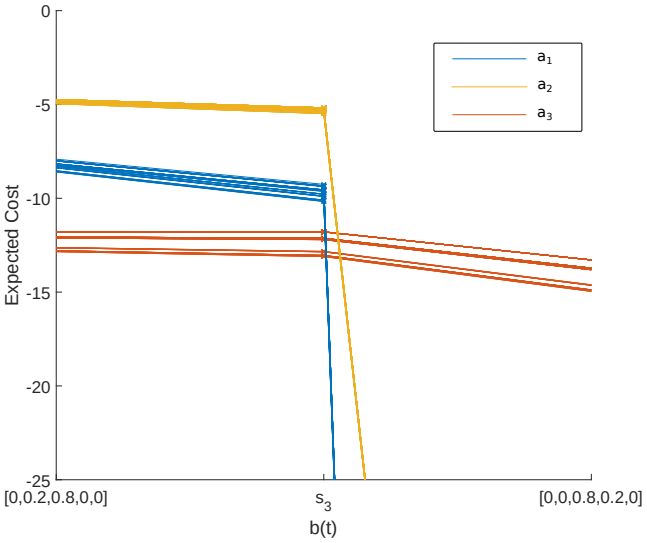

Figure 7.13. Value vector representation of the optimal policy in the range around $b(t) = s_3$, for U_1, $p_D = 0.8$ as developed by applying the incremental pruning algorithm.

With the threshold vector δ as set and discussed in Section 6.2 using the 75th percentile across simulations at $p_D = 1.0$, the performance monitoring scheme indicates model effectiveness for $p_D > 0.85$ as depicted in Figure 7.14. The scheme detects the discussed departure from optimal action selection via ϵ_a for $p_D < 0.85$ as depicted in Figure 7.14(b). The other two metrics, ϵ_ω and ϵ_χ, remain below threshold throughout the range of p_D explored as displayed in Figures 7.14(a) and (c) respectively.

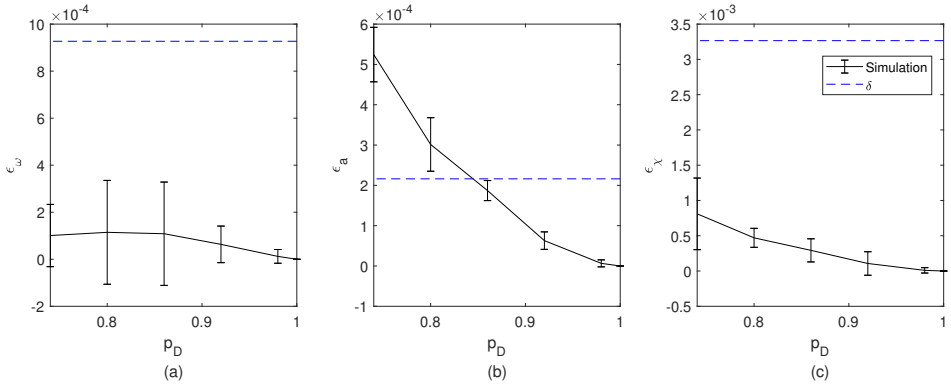

Figure 7.14. Error ϵ during simulated system operation as p_D degrades for U_3.

Although the decision making in s_3 is sub-optimal, the system errs on the side of increased caution on the part of the defender, resulting in greater attack suppression than was expected. Further, because the system spends far more time in the other states in which decisions are optimal, the sub-optimality in s_3 has little impact on the overall optimality of the system as it loses less than four percentage points in availability from what is expected. Thus, we consider the proposed system validated in that it performs as designed, implementing MTD to thwart attackers effectively while minimizing overhead.

7.2 Improvement over a Comparable System

We compare the proposed system to an alternative MTD system to quantify the gains possible under our scheme. In this case, the alternative system is attacker-agnostic and triggers reconfigurations pseudo-randomly with exponentially distributed inter-arrival times, similar to the defender with the two reconfigurations implemented in our validation system described in [8]. We use that as the basis for quantifying theoretical attack suppression and availability under the state-of-the-art-system. Preliminary results from this analysis were presented previously [80]. This section offers significant expansion of those early results.

Given the properties of exponential probability distributions, the probability of reconfiguration a_i occurring in a given time period t is

$$\psi_i = e^{(-\lambda_i t)}$$

given a rate of arrivals per second λ_i. Both a_2 and a_3 co-occurring describes a fourth action a_4 not considered in our proposed system, which carries probability

$$\psi_4 = e^{-t(\lambda_2+\lambda_3)}.$$

Consequently, the likelihood of a_1 (no action) is defined as

$$\psi_1 = 1 - \left(e^{(-\lambda_2 t)} + e^{(-\lambda_3 t)} - e^{-t(\lambda_2+\lambda_3)}\right).$$

These probabilities are used to define χ_R, system availability under the state-of-the-art-system as a sum of availability under each action weighted by their probabilities, i.e.,

$$\chi_R = \psi_1\chi_1 + (\psi_2 - \psi_4)\,\chi_2 + (\psi_3 - \psi_4)\,\chi_3 + \psi_4\chi_3$$

assuming that the loss of availability is concurrent when both a_2 and a_3 occur simultaneously.

These probabilities can also be used to define P_R, the transition probabilities of a Markov chain describing attack state under this system by taking a weighted sum as $\sum_{i=1}^{4} \psi_i P_i$ where ψ_i is the probability that each action occurs. We define matrix P_4 to represent state transition probability given the co-occurrence of both a_2 and a_3 with each element following

$$p_{4,i,j} = \frac{p_{2,i,j}p_{3,i,j}}{\sum_{j=1}^{n} p_{2,i,j}p_{3,i,j}}.$$

The state transition probability matrix under the state-of-the-art-system P_R is defined as weighted sum

$$P_R = \psi_1 P_1 + (\psi_2 - \psi_4)\,P_2 + (\psi_3 - \psi_4)\,P_3 + \psi_4 P_4,$$

which describes an absorbing Markov chain and thus defines the attack suppression under the state-of-the-art-system ϕ_R as

$$\phi_R = 1 - \frac{\tau_1}{\tau_R}$$

where τ_R is calculated in accordance with Equation 2.7.

By exploring a range of values for λ_2 and λ_3, it is possible to identify system parameters

wherein the state-of-the-art-system would match the attack suppression performance of the proposed system. We calibrated λ_2 and λ_3 so that $\phi_R = \phi$ and χ_R is maximized. Here, ϕ represents the attack suppression under the proposed system over the range $0.74 \leq p_D \leq 1.0$ as developed in Section 7.1.3. The calibration values for λ_2 and λ_3 and resultant availability under each case are compiled in Table 7.10.

Table 7.10. Comparison between the proposed system and a state-of-the-art-system calibrated to match in attack suppression under uncertainty profile U_3.

p_D	Proposed System		State-of-the-Art System			$\chi - \chi_R$
	ϕ (%)	χ (%)	λ_2	λ_3	χ_R (%)	
0.74	99.5	93.7	0.435	0.085	42.4	51.3
0.80	99.7	94.7	0.435	0.105	35.5	59.2
0.86	99.8	95.4	0.450	0.120	31.1	64.3
0.92	99.8	96.4	0.350	0.130	28.5	67.9
0.98	99.9	97.4	0.380	0.165	21.3	76.1
1.00	99.9	97.8	0.445	0.185	18.2	79.6

In all cases, the proposed system maintained greater availability than the state-of-the-art-system as indicated by the positive percentage point gains displayed in Figure 7.15. The worst case explored still offered a gain of more than 45 percentage points in availability. As p_D increases, the proposed system gains approach 80 percentage points. These results underscore the utility of our attack-sensing approach toward cost-controlled MTD.

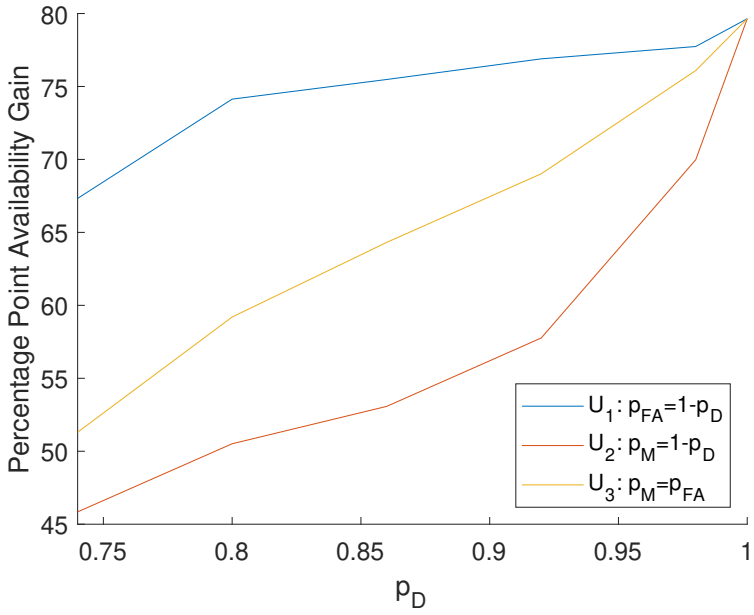

Figure 7.15. Percentage point availability gain ($\chi - \chi_R$) under the proposed system as compared to a comparable state-of-the-art-system calibrated to match in attack suppression.

7.3 System Performance Under Model Error

We concede that the performance of a model-based defensive system is dependent on the accuracy of the model, which is why it is important to quantify how the system performs under such conditions before considering the concept valid. These issues are explored in this section.

Error is possible in every aspect of the model, which precludes exhaustively testing the impact of every possible error over the complete range of possible values; however, the exploration of errors in cost matrix set C, observation probability matrix set O, and state transition probability matrix set P for the validation model presented in this section supports the system having acceptable error tolerance and error detection performance across the most likely and most dangerous error types.

7.3.1 Error Tolerance

The proposed system was tolerant during testing of small (< 5%) errors in O, P_1, and C with no significant change in availability or attack suppression performance. The summary of perturbation conditions identified during validation is detailed in Table 7.11.

Table 7.11. Model-error tolerance and impact within the proposed system. Tolerances presented are specific to the validation study.

Error Source	Description	Tolerance	Impact		
			χ	ϕ	optimality
Defense Analysis	c_{def}	±0.05	✓		✓
IDS Assessment	p_D	±0.05	✓		✓
Attack Analysis	$p_{1,i,i} \leftrightarrow p_{1,i,i+1}$	±0.20	✓	✓	
Attack Analysis	$P_1 + \theta P_{sys}$	$\theta < 0.33$	✓	✓	

Errors in cost matrix set C result from incorrectly measuring overhead of defenses in determining c_{def}. Intuitively, under testing, when the values in c_{def} are larger than the true overhead of the defenses, the impact is a gain of availability over the expected. If the values in c_{def} are too small, availability is reduced. In either case, attack suppression remains the same. Thus, optimization fails. These results were first introduced in Section 6.1.

Error in observation matrix set O stems from error in p_D via incorrect quantification during analysis of the intrusion detection system. As discussed in Chapter 6, within the validation model, error of up to 5% in p_D is well tolerated with overall system performance not impacted. The results of simulated operation of the system under flawed p_D were presented in Figure 6.9. Errors in O of up to 5% are well tolerated with ϵ staying below δ; however, beyond 5%, system performance degrades requiring improvement in the POMDP formulation before the system performance goals are achieved. The impact of these errors on the underlying action selection frequencies f_a are displayed in Figure 7.16. The tolerance in ϵ of up to 5% is repeated again here as f_a remains almost flat at this point. Beyond this boundary, however, increase in p_D error decreased the portion of time that the system fielded a_1. Thus, error in p_D resulted in increased overhead but not decreased attack suppression. The failure is in optimization, not defense. This is a desirable failure mode for this type of

system.

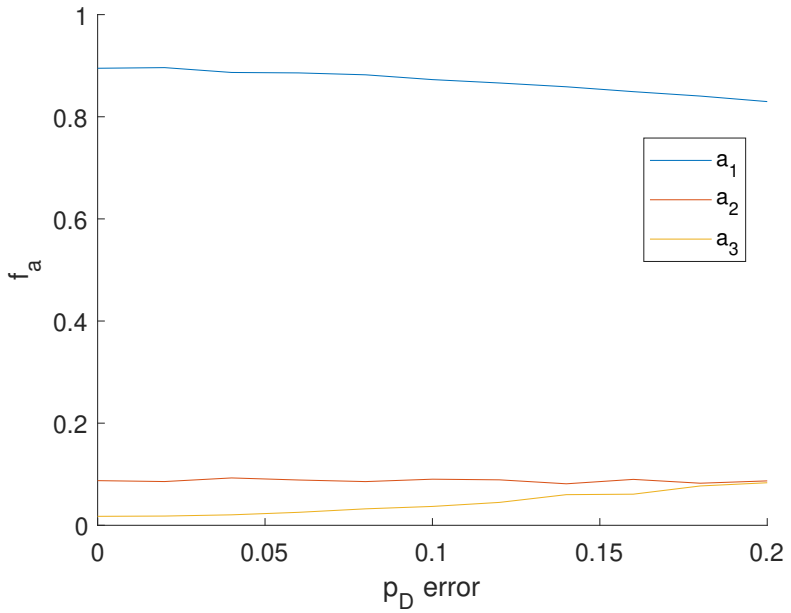

Figure 7.16. Frequency of action selection f_a begins to change as error in p_D increases beyond 5%.

To test the impact of errors in P_1, we analyzed system operation under imposed error magnitude in the range $[-0.2, 0.2]$ imposed on $p_{1,i,i}$ with the error displaced to $p_{1,i,i+1}$. Only the most extreme of valuse in the range result in a change in the optimal policy π found via Markov decision process theory. Thus, the system is tolerant of errors of this type up to this level. Attack suppression and availability depart from the expected values because the underlying dynamics are different, but the system continues to provide optimal attack suppression and availability.

We also tested the case in which the attacker is more systematic than expected such that

$$P_{SYS} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The systematic attacker was added to P_1 by an increasing weight factor θ so that the resultant attack dynamics P_1' follow

$$P_1' = \theta P_{SYS} + (1 - \theta)P_1.$$

In this case, the optimal policy is stagnant for $\theta < 0.33$, which develops a boundary value beneath which error of this type is well tolerated by the proposed system under the context of the validation scenario.

While there are a multitude of potential errors in a model of this complexity, exploration of the validation model has determined that the proposed system is tolerant of many of the broad error classes expected.

7.3.2 Error Indication via the Performance Monitoring Scheme

Beyond the boundaries of error tolerance, the performance monitoring scheme was introduced to flag model error as rapidly as possible. Errors are measured during system operation and compiled into error vector $\epsilon = [\epsilon_\omega, \epsilon_a, \epsilon_\chi]$.

Baseline for ϵ

First, to establish the validity of these error metrics, we ensured that ϵ is indeed stable and close to zero when the system is operating under ideal conditions wherein there is perfect model effectiveness. Model effectiveness is the measure of how well the POMDP represents the operational environment. To support our conclusion that these error metrics are valid, the means and 15th and 85th quantile of each error by step are displayed in Figure 7.17. The individual components that contribute to this stability were discussed in Section 6.2.

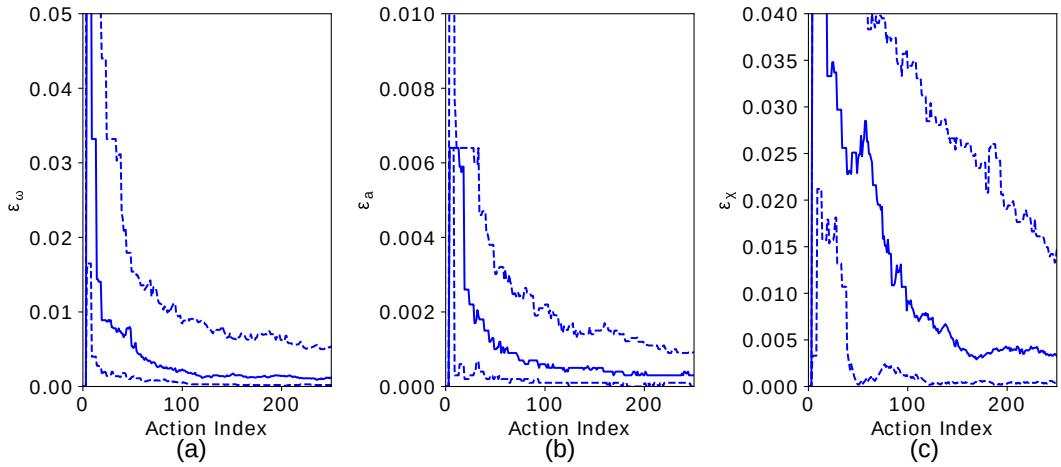

Figure 7.17. Trends in ϵ over 100 trials of the first 250 decisions under proposed system, POMDP known to be valid, attacker profile drawn from [17], $p_D = 1.0$. Used as *control* in subsequent error tests.

Detection of P_1 Error

We reviewed the impact on error metrics when P_1 contains significant error, imposed in two cases. In the first case, attack intensity abruptly changes at step 2000, taking on a far less aggressive profile following

$$P_1' = 0.25P_1 + \begin{bmatrix} 0.75 & 0 & 0 & 0 & 0 \\ 0.75 & 0 & 0 & 0 & 0 \\ 0.75 & 0 & 0 & 0 & 0 \\ 0.75 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.75 \end{bmatrix} \tag{7.2}$$

In the second case, the attacker abruptly becomes more aggressive after step 2000 following

$$P_1' = 0.25P_1 + \begin{bmatrix} 0 & 0.75 & 0 & 0 & 0 \\ 0 & 0 & 0.75 & 0 & 0 \\ 0 & 0 & 0 & 0.75 & 0 \\ 0 & 0 & 0 & 0 & 0.75 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.3}$$

such that the attacker is now 75% more likely to push forward to the next state than is expected by the system.

Both cases are detected by the performance monitoring scheme as ϵ increases after either profile change as depicted in Figure 7.18–7.19 for the less and more aggressive cases respectively. Thus, these errors would each result in POMDP reformulation. It also appears that either attacker profile change manifests differently in ϵ such that model error classification might be achieved in future work.

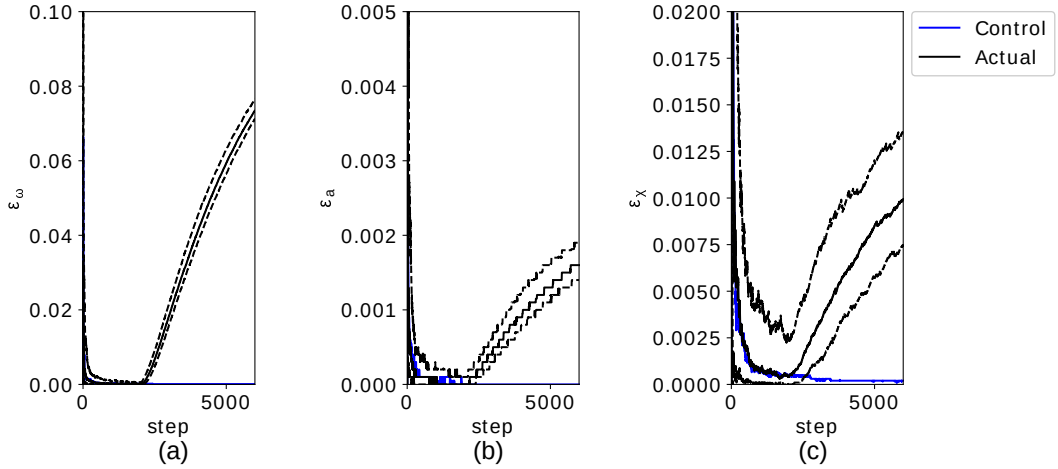

Figure 7.18. Trends in ϵ over 100 trials of the first 6000 decisions under proposed system, less aggressive attacker profile following Equation 7.2, $p_D = 1.0$.

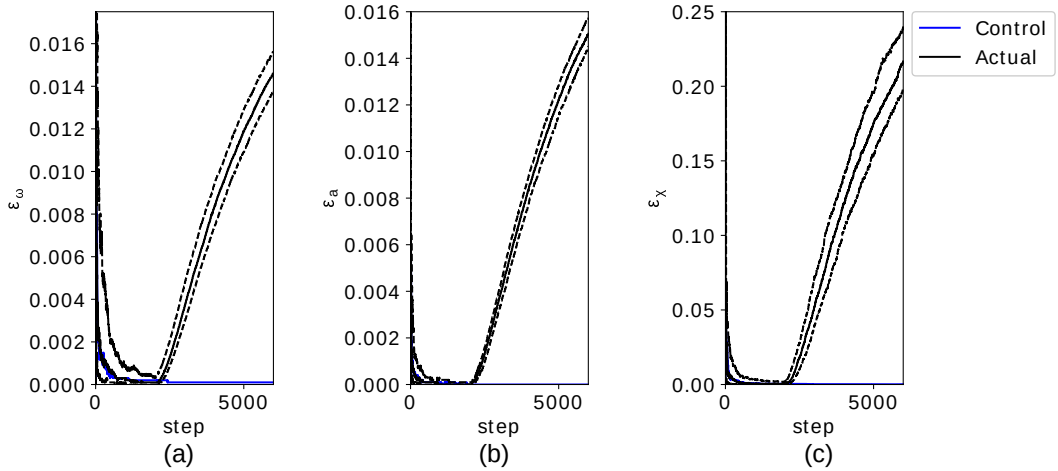

Figure 7.19. Trends in ϵ over 100 trials of the first 6000 decisions under proposed system, more aggressive attacker profile following Equation 7.3, $p_D = 1.0$.

Detection of Failed Defenses

Failed defenses manifest in the system as P_i model errors for $1 < i \leq m$. We tested the most extreme P_i error wherein defenses fail such that $P_i \equiv P_1$. In the first test, the failure was imposed at initialization. The performance monitoring scheme rapidly detects the issue with ϵ failing to approach the expected values as displayed in Figure 7.20. While the detection is rapid, equally rapid intervention would be required to repair the defenses; otherwise, attack goal state s_5 will be reached within 3 minutes as discussed in Section 7.1.1. Indeed, $\epsilon \rightarrow \epsilon_{atk}$, indicating that the system was successfully attacked across all 100 trials.

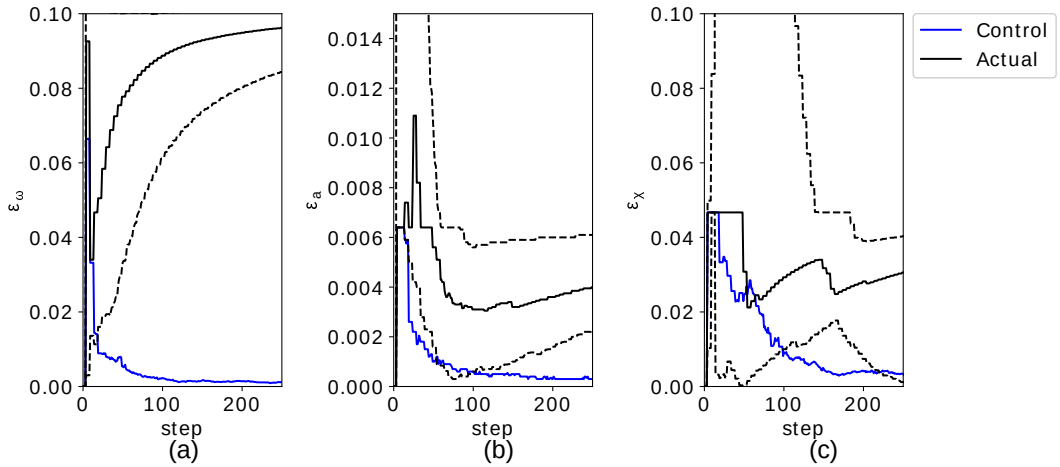

Figure 7.20. Trends in ϵ over 100 trials of the first 250 decisions under proposed system, a_2 and a_3 failure from initialization, attacker profile drawn from [17], $p_D = 1.0$. The mean is plotted as a solid line, while the 15th and 85th quantile are plotted as dashed lines.

We also tested delayed onset defensive failure that might occur either if there is a technical glitch in the defense or if the attacker was able to gain enough insight into the mechanism of MTD as to render it ineffective. In this case, the system operated with model effectiveness through 2,000 decisions before a_2 failed such that $P_2 \equiv P_1$. Again, detection is rapid, with steep increases in ϵ_a and ϵ_χ in particular, as displayed in Figure 7.21.

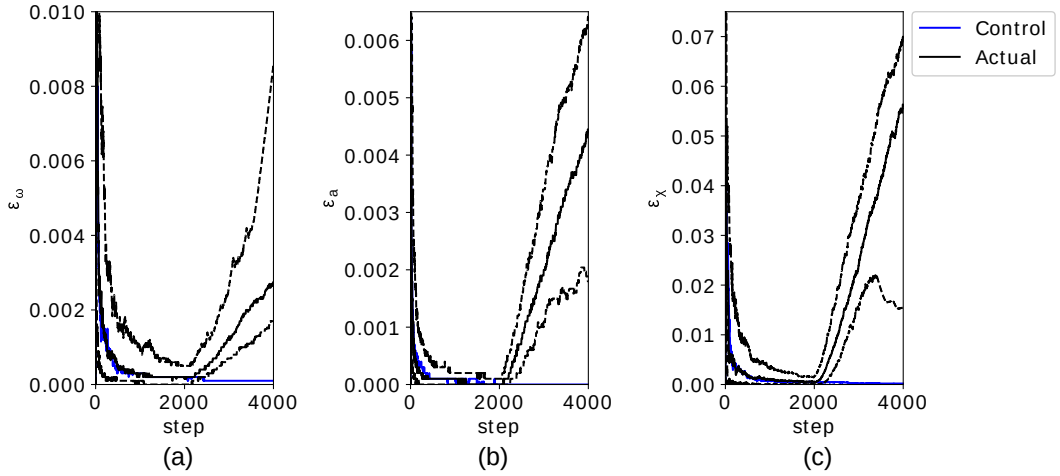

Figure 7.21. Trends in ϵ over 100 trials of the first 4000 decisions under proposed system, a_2 failure onset at step 2000, attacker profile drawn from [17], $p_D = 1.0$. The mean is plotted as solid line, while the 15th and 85th quantile are plotted as dashed lines.

In all cases, detection of error by the performance monitoring scheme results in a return to POMDP formulation. The model must be corrected via the same attack and defense analysis, now informed by the error indications and additional data available relating to attack and defense dynamics.

7.4 Expanded Dimensionality

The core of our validation study implements a five-state, three-action model with dimensionality of $|s| \times |\omega| \times |a| = 75$. We also conducted cursory exploration of system performance under conditions of greater dimensionality. Our expanded model contains nine attack phases, nine corresponding observations, and four actions for a total dimensionality of $|s| \times |\omega| \times |a| = 324$.

In this study, our focus was confirming that additional model capacity existed. As such, the nine-state, four-action model explores a notional attack-defense process because specific forensic data similar to that levied for the five-state attack model could not be found. Instead, we consider a systematic attack process that visits nine phases in order moving

forward one state at each transition such that

$$P_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The nine states represent a more sophisticated attack wherein the attacker strings together two exploits to reach an ultimate goal. States s_1 through s_6 represent the preliminary process *exploit a*, and s_7 through s_9 represent a secondary process, *exploit b*. We established state transition probabilities and costs under each available defense to ensure that scaling defensive response to imminence of attack risk was possible while also considering that a single defense may not disrupt multiple exploit processes.

Defense a_2 is low cost, but also of limited utility, analogous to a TCP connection reset or similar simple reconnaissance disruption. Once the attacker completes the first exploit process, a_2 becomes obsolete. It carries an overhead of $c_2 = -0.1$ and returns the attacker to s_1 with $p_s = 0.5$. The next defense a_3 carries a cost of $c_3 = -1$ and is effective in states s_3 through s_8, returning the attacker to s_3 with $p_s = 0.7$ to reset the foothold gained by *exploit a*. The final option a_4 carries a cost of $c_4 = -10$ and impacts states s_7 and s_8 disrupting the success of the launch of *exploit b*. This defense returns the system to s_7 with $p_s = 0.9$.

The parameters applied for each defense are complied in Table 7.12. Additionally, we list availability and attack suppression in Table 7.12. Attack suppression was calculated assuming defense a_i is used alone and repetitively as though operating with policy vector $\pi = [a_i, a_i, ..., a_i]$.

Table 7.12. Notional actions available in expanded model

Action	Description	p_s	States Impacted	c_i	Attack Suppression
a_1	no action	n/a	n/a	0.0	0.00%
a_2	reconnaissance disruption	0.5	$s_1 : s_6$	-0.1	87.69%
a_3	*exploit a* disruption	0.7	$s_3 : s_8$	-1.0	97.35%
a_4	*exploit b* disruption	0.9	$s_7 : s_8$	-10.0	93.10%

We apply attack penalty scaling factor $\nu = 100$ such that $c_{atk} = -1000$. We simulated system operation under $\gamma = 0.75$ over 10 trials lasting 6,000 decisions each per test point. Test points were under uncertainty model U_3 at $p_D = [0.80, 0.85, 0.90, 0.95, 1.0]$. Based on analysis of P_1, we implemented the MTD selection subsystem with parameters computational time $z = 10$, particle count $q = 100$, and depth $d = 8$.

The proposed system accomplished better than 94% attack suppression while maintaining an overhead-per-decision of better than -0.9 even at $p_D = 0.80$. Additionally, performance in simulation came within 5% of the expected attack suppression and 0.7 of the expected overhead-per-decision. Expected metrics were determined via Equations 6.3 and 6.5, respectively. Attack suppression performance at each test point is compiled in Table 7.13 while overhead-per-decision is presented in Table 7.14.

At this dimensionality, the equivalent MDP has $n^m = 9^4$ possible policy vectors ranging in discounted expected total value from a low of -400 to a high of -0.64. This latter value is exhibited by the true optimal policy vector π, but there is a tight cluster of 32 near-optimal policies with discounted total values within 0.50 of that for π. Every simulated decision delivered by the proposed system came from within this tight cluster of near-optimal policies, but only a handful came from the true optimal policy that is used to determine the expected metrics. The proposed system was unable to distinguish the true optimal decision from the near-optimal neighbors, which is why the proposed system under-performed the expected metrics in this scenario.

Driving these near-optimal decisions closer to alignment with π and thus achieving perfor-

mance nearer to the expected metrics is possible by adjusting MTD subsystem parameters z, q, and d as these parameters control a trade-off between computational investment and optimality of each decision [24]. Still, the delivery of consistently near-optimal decisions under the expanded model validates that excess capacity is available. The proposed system can be used to deliver overhead minimization for an MTD system even as POMDP dimensionality increases.

Table 7.13. Attack suppression (%) under the expanded model

	p_D				
	1.00	0.95	0.90	0.85	0.80
Expected	99.85	99.78	99.71	99.64	99.57
Mean	99.00	96.73	96.83	96.26	94.98
Minimum	97.31	85.19	85.45	81.82	86.44
Maximum	99.86	99.81	99.78	99.80	99.69

Table 7.14. Overhead-per-decision under the expanded model

	p_D				
	1.00	0.95	0.90	0.85	0.80
Expected	-0.1537	-0.1558	-0.1579	-0.1602	-0.1627
Mean	-0.3305	-0.5216	-0.4114	-0.4113	-0.8967
Minimum	-0.7477	-1.1964	-0.6353	-0.6760	-2.1567
Maximum	-0.2058	-0.2269	-0.1333	-0.1280	-0.1716

Although this exploration involved a deterministic cyber attack process and notional defenses, the model is four times greater in dimensionality than the model used to conduct our core research. As such, these results confirm that there is significant room for expansion of state, observation, and action spaces in the POMDP at the core of the proposed system. Handling more complex attacks with a greater variety of defensive techniques is possible.

7.5 Summary

This chapter has validated the proposed system through simulation in a realistic attack-defense context. The system provided optimal defense against a five stage attack via MTD. We have provided evidence that the system performs as designed and offers significant availability improvement over a comparable MTD system for the same attack suppression. We have also explored system performance in the presence of model error, both identifying tolerances below which the error is of little concern and providing examples of how errors would be detected to facilitate POMDP reformulation. Finally, we have confirmed that the system will be able to expand to incorporate more complex attack models and greater libraries of MTD techniques.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 8:
Conclusion

In this dissertation, we proposed and validated a mechanism by which MTD could be implemented optimally so that it is both effective in creating an unpredictable attack surface and manageable from the defensive perspective. We have identified and validated the components necessary to facilitate system operations and developed metrics for predicting and assessing performance. We have also leveraged those metrics for performance monitoring to prevent model error from leading to system failure.

We conducted our research through formulation of a stochastic model of attack-defense dynamics as a POMDP. We then conducted analysis of the properties of the model so that it could be applied to facilitate autonomous selection of MTD. From there, we developed a simulation of the proposed system and explored performance under ideal conditions before imposing various errors to determine how robust the proposed architecture is to both state and model uncertainty within the POMDP. We also conducted comparative analysis against a state-of-the-art MTD system to quantify the gains possible under the proposed system and confirmed that model expansion to consider more complex attack models and greater libraries of MTD techniques is possible.

Our results support the conclusion that the proposed system delivers effective attack suppression and overhead minimization within constraints of intrusion detection and model effectiveness. We put forth new best-practices for model formulation in support of optimal MTD and support POMDP in particular as a path for achieving MTD that is both effective and resource-conscious in thwarting attackers.

8.1 Contributions

Our results make several contributions toward more optimal and thus adoptable MTD. The key contributions relate to the formulation of POMDP for triggering defenses as needed and the development of a performance monitoring scheme to improve system performance.

We have demonstrated the utility of POMDP formulation as an objective basis by which

MTD effectiveness and overhead can be studied and is particularly useful in such analysis for systems where multiple MTD techniques are used in combination. The developed formulation is useful for the family of MTD that incur per-reconfiguration overhead, facilitating the trade-off analysis necessary to implement MTD so that cost and benefit are balanced. Our work has expanded the understanding of best practices for model formulation in support of MTD optimization. Examples include adopting attack models with intermediate phases represented and defining the impact of defenses as a function of the underlying attack model to permit rapid update to the system as attack patterns change. In addition to permitting static analysis of attack-defense dynamics, the model formulation drives autonomous optimal decision making within our proposed system.

We have also described a technique for scaling defenses to the proximity of threat via the combination of an absorbing Markov chain, an attack penalty scaled by defensive overhead, and depth limitations within Monte Carlo planning. The absorbing Markov chain and attack penalty ensure that most effective actions are selected in higher indexed states when attack risk justifies overhead expenditure. Meanwhile, the depth limitation ensures less costly and less effective defenses are viable in lower indexed states when attack is not imminent.

Our simulations have vetted the proposed system to determine best practices under realistic scenarios. The proposed system thwarted 99% of inbound attacks, which changes the rate of successful attacks from every three minutes to every 9.5 hours, all while sustaining system availability at greater than 94%. A comparable system that triggered reconfigurations randomly maintained just 43% availability in achieving this same attack suppression rate, which illustrates the utility of optimizing MTD in the way we have proposed.

Finally, we have combined elements from absorbing Markov chain and POMDP theory to facilitate predictive analysis of system performance under ideal and error conditions as part of a performance monitoring scheme unique in the field of POMDP-based cybersecurity controllers. For similar systems with an absorbing ultimate state s_n, state-observation alignment, observations independent of action selection, and neighboring uncertainty, the equations defined in Section 6.1 flag model error in time for corrective action, which is an important stride toward confident adoption of a model-based defensive system.

8.2 Future work

These contributions are poised for expansion in a few directions. The next steps relate to reducing the amount of manual analysis necessary to operate the system, expanding the complexity incorporated into the POMDP, and working toward practical adoptability.

8.2.1 Automated Model Updates

It would be useful to introduce automation into formulation and fine-tuning of the POMDP. Currently the system handles state uncertainty autonomously but only flags model uncertainty. Under conditions of state uncertainty, the exact state at any given observation point is not perfectly known while under model uncertainty, components including state transition probabilities, observation probability, and defensive costs are not certain [71]. Bayes-adaptive POMDP [83] may be the answer. In this case, after a starting guess, model parameters are updated via reinforcement learning techniques to adapt to the changing environment. Work has been conducted to develop reinforcement learning techniques in combination with POMDP to facilitate optimal decision making when both state and model uncertainty exist [74], [84] that may be a good fit for the proposed system.

Another path to increasing automation in POMDP formulation would be to incorporate error classification into the performance monitoring scheme. As currently designed, the system stops after detection of $\epsilon > \delta$, but there is indication that ϵ follows particular trends for each error type that could be exploited for error classification toward more rapid manual or even automated reformulation of the model. In general, the system would be improved via more formal inspection of the anomaly detection and classification conducted within the performance monitoring scheme.

Further, based on complimentary progress in applying machine learning for intrusion detection, an investigation should also be performed toward leveraging machine learning to automate and improve portions of the proposed system. In particular, artificial neural networks already under investigation toward intrusion detection and classification [85] might be extended to determine observation probability matrices O or track belief state $b(t)$.

8.2.2 Expanding Complexity

We limited our state, action, and observation space to facilitate direct comparison with an existing state-of-the-art MTD system. Now that we understand the gains possible, the system should be expanded to incorporate additional complexity. Specifically, the quantity and variety of both attacks and defenses included in the proposed system should be increased.

Greater dimensionality in terms of state, defense, and observation spaces would facilitate consideration of other attack progressions and new MTD combinations. Our detailed validation study implements dimensionality of $|s| \times |\omega| \times |a| = 75$. We also performed cursory analysis confirming system operation at $|s| \times |\omega| \times |a| = 324$. DESPOT has been verified to handle $|s| \times |\omega| \times |a| = 10^6$ [24]. Assuming that all 92 existing MTD [6] were incorporated into the system with one non-action so that $|a| = 93$ and constraint $s \equiv \omega$ holds, the proposed system could support $|s| \leq 104$, which would facilitate inclusion of other attack types against individual hosts or expansion toward network-wide optimal MTD.

Toward expanding variety, there are certainly more MTD given the mentioned catalog, but there are also other defensive mechanisms that could be incorporated that range in sophistication from physically disconnecting resources from the network up through cyber deception and decoying. The model might also be expanded to explore a hybrid system that optimally triggers defenses based on assessed attack risk but also includes a latency threshold to ensure a minimal reconfiguration rate is maintained.

8.2.3 Practical Adoptability

Finally, we have vetted this system under a variety of expected and realistic model parameters to prepare for practical implementation of a POMDP-based cyber defensive scheme, but this simulated operation is just the first step toward a practically adoptable system.

An investigation into specialized hardware for DESPOT decision making is an important next step as simulation-based optimization involves highly repetitive calculations. General purpose central processing units like those used in our simulations are not ideal for this task, but both field programmable gate arrays and graphics processing units have shown promise in similar applications [86], [87]. Either might improve the performance of the proposed system by reducing the computational time required for decisions.

Identifying the ideal upstream IDS to support the attack-phase-based observations required by the proposed system is also a point for future work. Although uncertainty-handling measures are at the core of the system, performance is still tied to the accuracy of intrusion detection such that work to identify the specific IDS architecture that best compliments the proposed system will only further improve our ability to accomplish optimized MTD.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A:
Validation Model

This appendix details the specific model parameters used in the validation study wherein the proposed system thwarts the five stage attack detailed in Figure A.1.

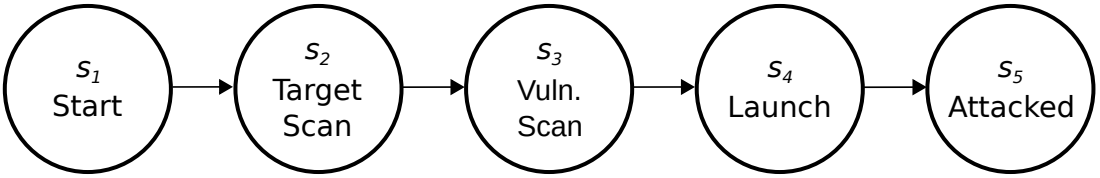

Figure A.1. Five phase attack process used for validation of proposed system

A POMDP is fully described by $\{s, a, \omega, P, C, O, \gamma\}$. For the validation study, the following parameters were used: States are aligned with each of the five attack stages such that

$$s = [s_1, s_2, s_3, s_4, s_5].$$

Actions are drawn from a non-action, a service reconfiguration, and an IP address reconfiguration such that

$$a = [a_1, a_2, a_3].$$

Observations from the upstream IDS are aligned with the possible states such that $\omega \equiv s$ and thus

$$\omega = [s_1, s_2, s_3, s_4, s_5].$$

State transition probabilities in the absence of defense, P_1, were determined from honeypot data published in [17] yielding

$$P_1 = \begin{bmatrix} 0.6611 & 0.2300 & 0.0856 & 0.0233 & 0 \\ 0 & 0.9235 & 0.0687 & 0.0078 & 0 \\ 0 & 0 & 0.7900 & 0.2100 & 0 \\ 0 & 0 & 0 & 0.5000 & 0.5000 \\ 0 & 0 & 0 & 0 & 1.0000 \end{bmatrix}.$$

State transition probabilities under either defense, P_2 or P_3 were obtained as a function of P_1. For a_2, we implement a service reconfiguration for which the state transition probability matrix is given as

$$P_2 = \begin{bmatrix} \frac{\mu-\mu_v}{\mu} & 0 & 0 & 0 & 0 \\ \frac{\mu-\mu_v}{\mu} & 0 & 0 & 0 & 0 \\ \frac{\mu-\mu_v}{\mu} & 0 & 0 & 0 & 0 \\ \frac{\mu-\mu_v}{\mu} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{\mu-\mu_v}{\mu} \end{bmatrix} + \frac{\mu_v P_1}{\mu}.$$

With the specific values used in validation,

$$P_2 = \begin{bmatrix} 0.8870 & 0.0767 & 0.0285 & 0.0078 & 0 \\ 0.6667 & 0.3078 & 0.0229 & 0.0026 & 0 \\ 0.6667 & 0 & 0.2633 & 0.0700 & 0 \\ 0.6667 & 0 & 0 & 0.1667 & 0.1667 \\ 0 & 0 & 0 & 0 & 1.0000 \end{bmatrix}.$$

For a_3, we implement an address reconfiguration such that

$$P_3 = \begin{bmatrix} \frac{\rho-1}{\rho} & 0 & 0 & 0 & 0 \\ \frac{\rho-1}{\rho} & 0 & 0 & 0 & 0 \\ \frac{\rho-1}{\rho} & 0 & 0 & 0 & 0 \\ \frac{\rho-1}{\rho} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{\rho-1}{\rho} \end{bmatrix} + \frac{P_1}{\rho}.$$

Given an address space of $\rho = 256$ options, the specific state transition probabilities follow as

$$P_3 = \begin{bmatrix} 0.9987 & 0.0009 & 0.0003 & 0.0001 & 0 \\ 0.9961 & 0.0036 & 0.0003 & 0.0000 & 0 \\ 0.9961 & 0 & 0.0031 & 0.0008 & 0 \\ 0.9961 & 0 & 0 & 0.0020 & 0.0020 \\ 0 & 0 & 0 & 0 & 1.0000 \end{bmatrix}.$$

These defenses have state-independent availability costs, $c_2 = -0.635$ and $c_3 = -9.59$. The attack penalty is set at $c_{atk} = \nu \min[c_{def}] = 100c_3 = -959$ so that combined costs yield the cost matrix

$$C = \begin{bmatrix} c_1 & c_2 & c_3 \\ c_1 & c_2 & c_3 \\ c_1 & c_2 & c_3 \\ c_1 & c_2 & c_3 \\ c_1 + c_{atk} & c_2 + c_{atk} & c_3 + c_{atk} \end{bmatrix}$$

with specific values applied yielding

$$C = \begin{bmatrix} 0 & -0.635 & -9.590 \\ 0 & -0.635 & -9.590 \\ 0 & -0.635 & -9.590 \\ 0 & -0.635 & -9.590 \\ -959 & -959.635 & -968.590 \end{bmatrix}.$$

Partial observability was explored over a range of detection probabilities, $0.75 \leq p_D \leq 1.0$, under uncertainty conditions U_1, U_2, and U_3 that result in observation matrices

$$O_{U_1} = \begin{bmatrix} p_D & 1 - p_D & 0 & 0 & 0 \\ 0 & p_D & 1 - p_D & 0 & 0 \\ 0 & 0 & p_D & 1 - p_D & 0 \\ 0 & 0 & 0 & p_D & 1 - p_D \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$O_{U_2} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1-p_D & p_D & 0 & 0 & 0 \\ 0 & 1-p_D & p_D & 0 & 0 \\ 0 & 0 & 1-p_D & p_D & 0 \\ 0 & 0 & 0 & 1-p_D & p_D \end{bmatrix},$$

and

$$O_{U_3} = \begin{bmatrix} p_D & 1-p_D & 0 & 0 & 0 \\ \frac{1-p_D}{2} & p_D & \frac{1-p_D}{2} & 0 & 0 \\ 0 & \frac{1-p_D}{2} & p_D & \frac{1-p_D}{2} & 0 \\ 0 & 0 & \frac{1-p_D}{2} & p_D & \frac{1-p_D}{2} \\ 0 & 0 & 0 & 1-p_D & p_D \end{bmatrix},$$

respectively. Finally, we set $\gamma = 0.75$ to ensure both attack suppression and availability are considered in balance.

APPENDIX B:
Additional Results

This appendix includes additional results from the trials of the proposed system. First, overall attack suppression and availability are presented for U_1 and U_2 in Figures B.1 and B.2, respectively. Similar to the results for U_3 discussed in Chapter 7, the system suppresses over 99% of attacks while maintaining availability of 94% in both cases. Action visits under the alternate cases followed the trends of U_3 as the system selects a_3 more than expected here too. These results are presented in Figures B.3 and B.4. Examination of the results for cases U_1 and U_2 was helpful in identifying the reflection of the impact of uncertainty described in Section 6.1.

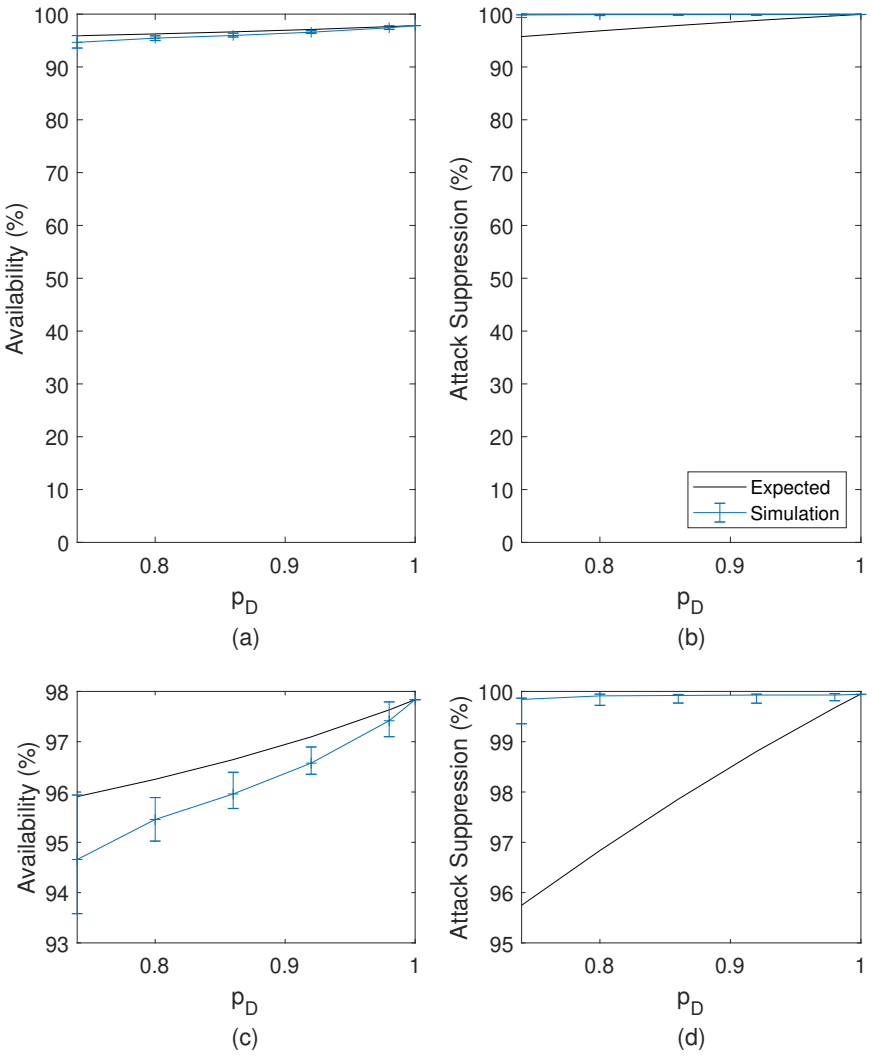

Figure B.1. Performance metrics for case U_1. The solid black line indicates predicted performance based on Equations 6.3 and 6.5, while the blue line represents the mean across simulated operation of the proposed system, with the 0.25 and 0.75 quantile indicated by the error bars.

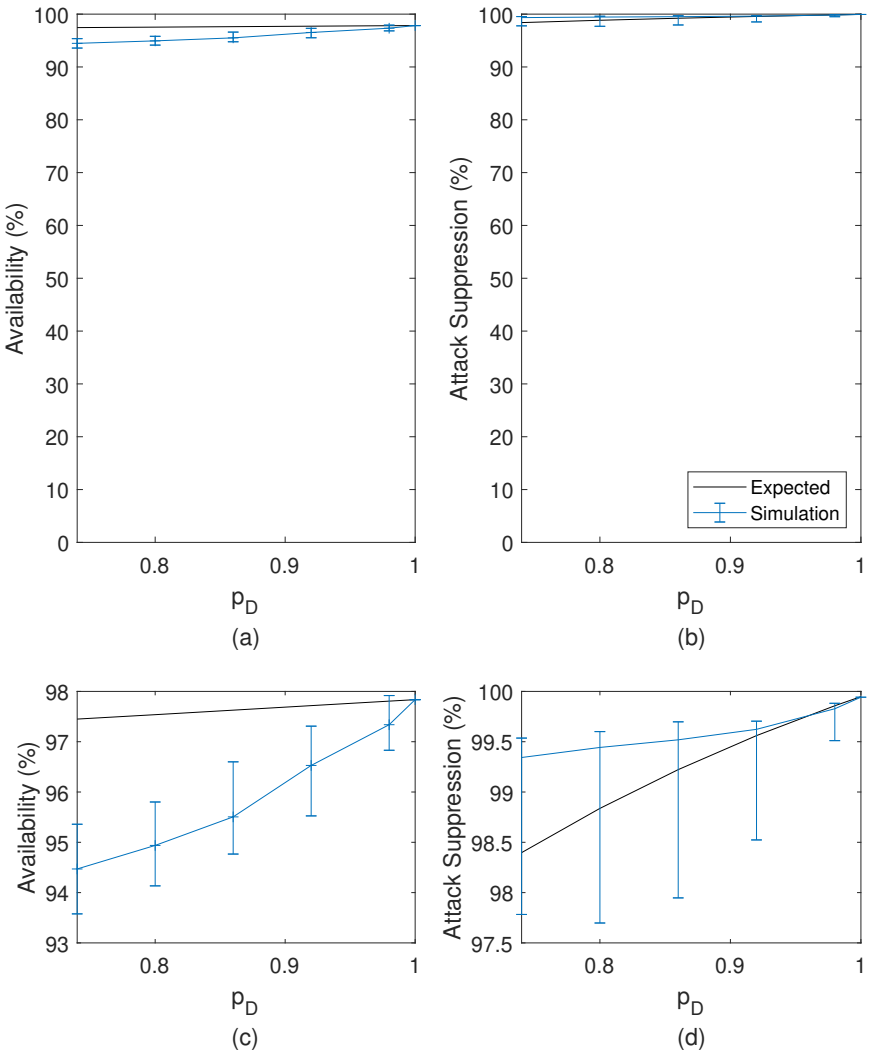

Figure B.2. Performance metrics for case U_2. The solid black line indicates predicted performance based on Equations 6.3 and 6.5, while the blue line represents the mean across simulated operation of the proposed system, with the 0.25 and 0.75 quantile indicated by the error bars.

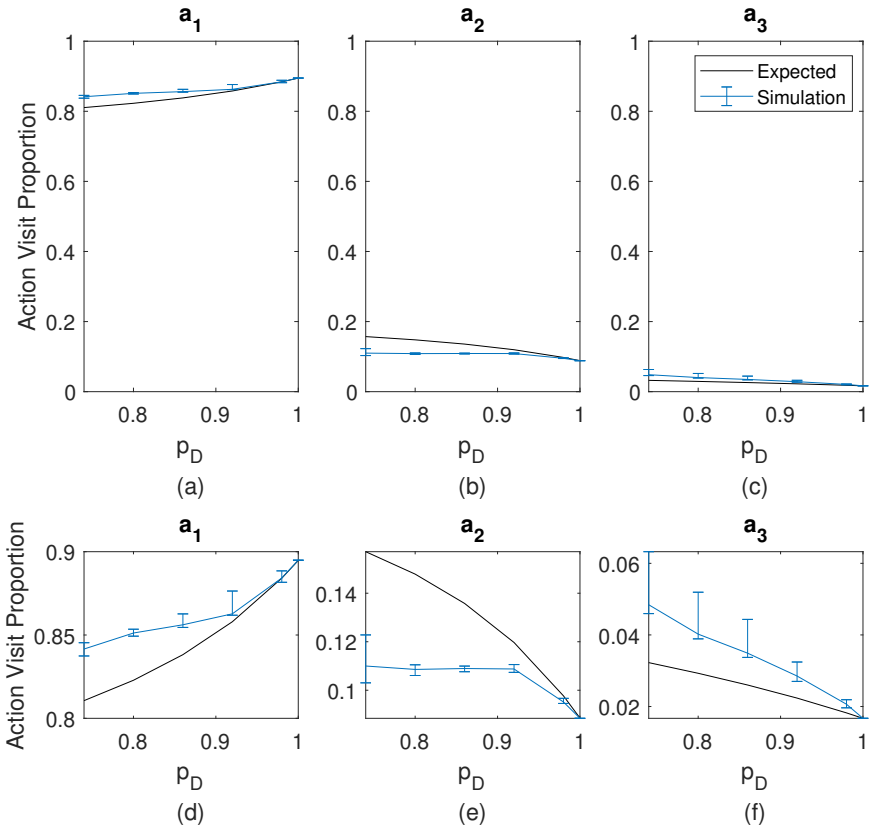

Figure B.3. Action visit trends for case U_1 The solid black line indicates predicted performance based on Equation 6.4, while the blue line represents the mean across simulated operation of the proposed system, with the 0.25 and 0.75 quantile indicated by the error bars.

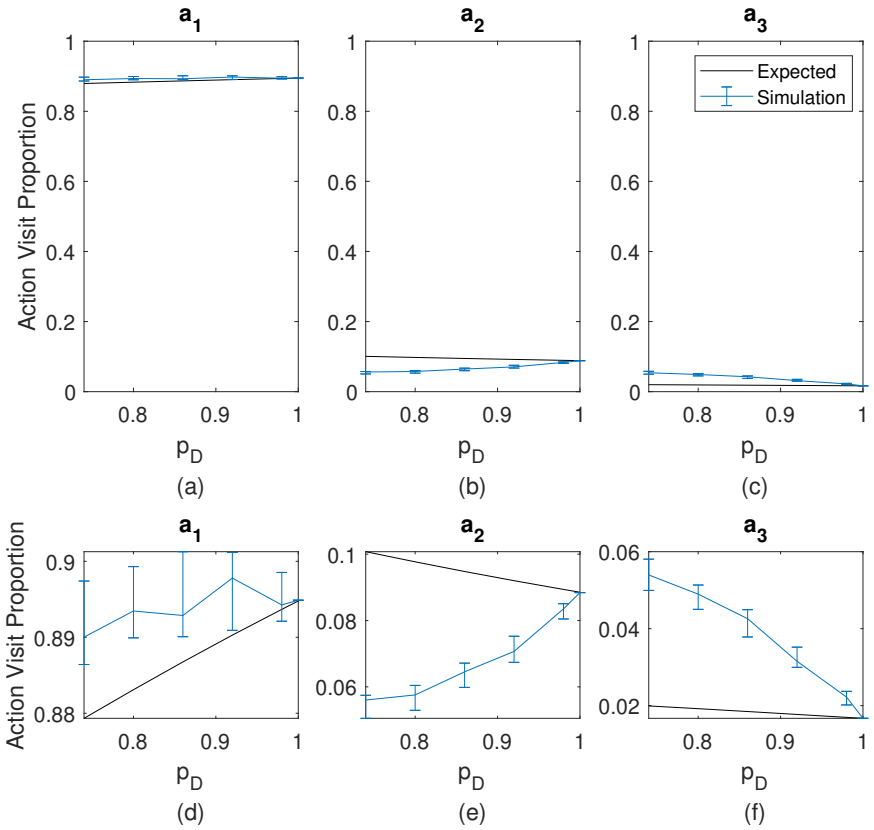

Figure B.4. Action visit trends for case U_2 The solid black line indicates predicted performance based on Equation 6.4, while the blue line represents the mean across simulated operation of the proposed system, with the 0.25 and 0.75 quantile indicated by the error bars.

Alignment between expected and actual state and observation frequencies support the same conclusions detailed in Chapter 7 and were thus not included in discussion. We include plots of the specific results here. The alignment between state frequency of occurrence with the expected values calculated via Equation 6.1 is displayed in Figures B.5, B.6, and B.7.

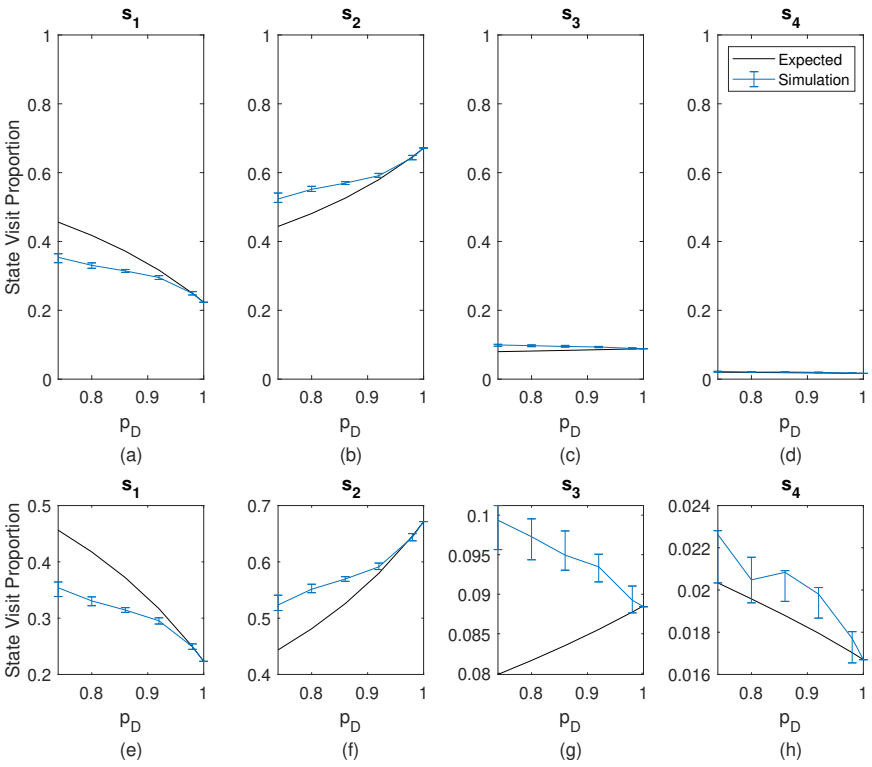

Figure B.5. State visit trends for case U_1 The solid black line indicates predicted performance based on Equation 6.1, while the blue line represents the mean across simulated operation of the proposed system, with the 0.25 and 0.75 quantile indicated by the error bars.

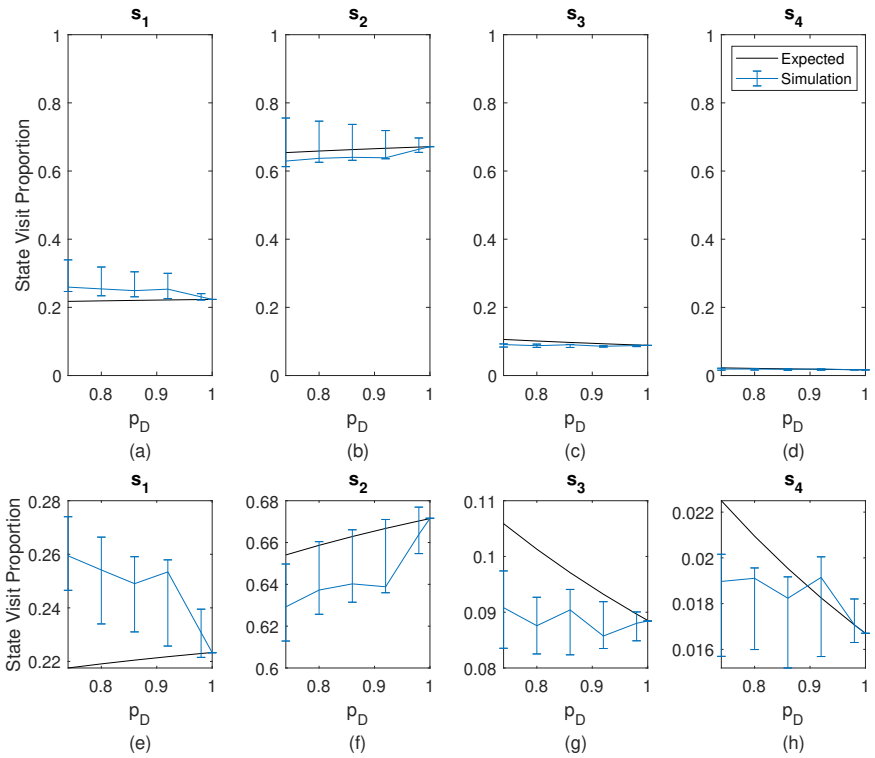

Figure B.6. State visit trends for case U_2 The solid black line indicates predicted performance based on Equation 6.1, while the blue line represents the mean across simulated operation of the proposed system, with the 0.25 and 0.75 quantile indicated by the error bars.

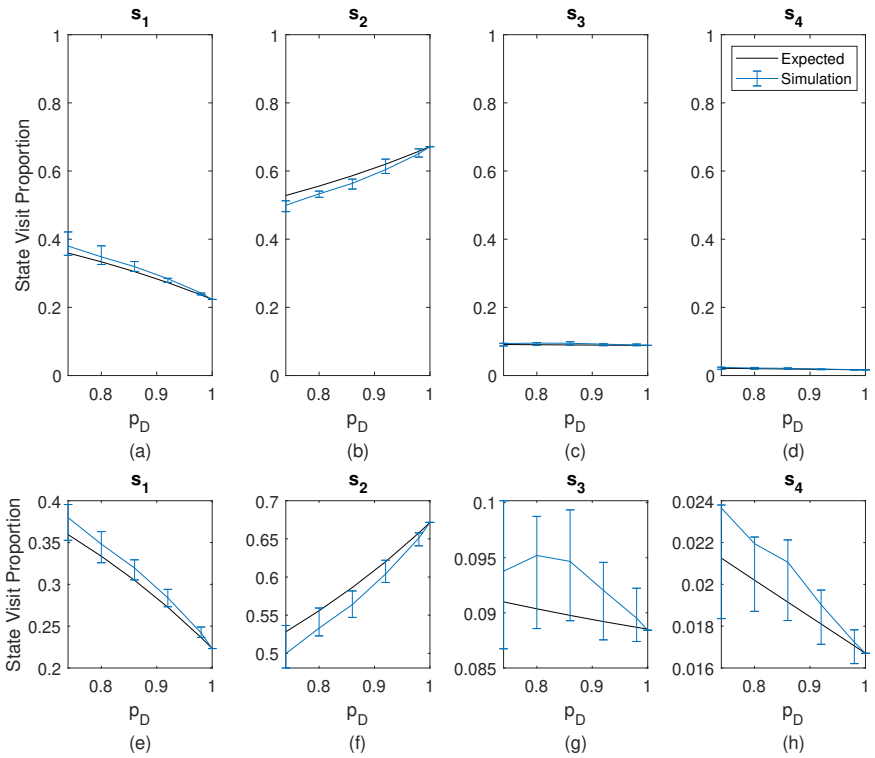

Figure B.7. State visit trends for case U_3 The solid black line indicates predicted performance based on Equation 6.1, while the blue line represents the mean across simulated operation of the proposed system, with the 0.25 and 0.75 quantile indicated by the error bars.

The alignment between observation frequency of occurrence with the expected values calculated via Equation 6.2 is displayed in Figures B.8, B.9, and B.10. .

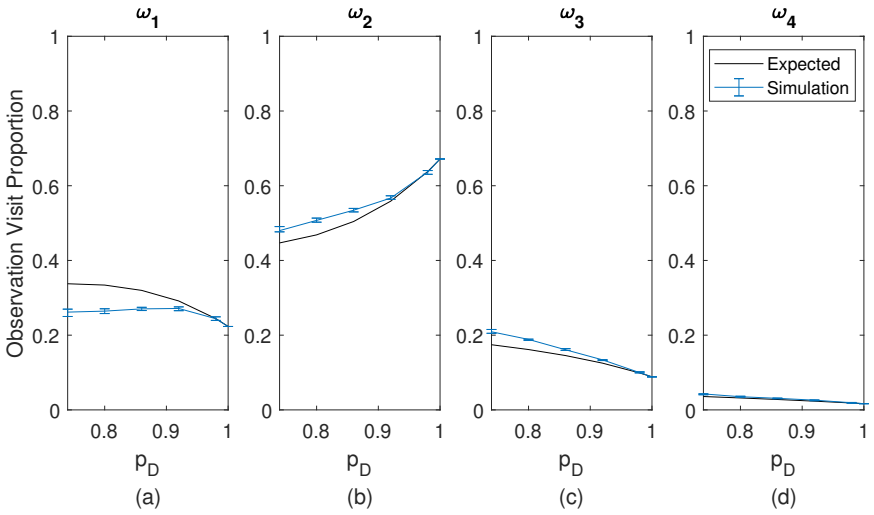

Figure B.8. Observation visit trends for case U_1 The solid black line indicates predicted performance based on Equation 6.2, while the blue line represents the mean across simulated operation of the proposed system, with the 0.25 and 0.75 quantile indicated by the error bars.

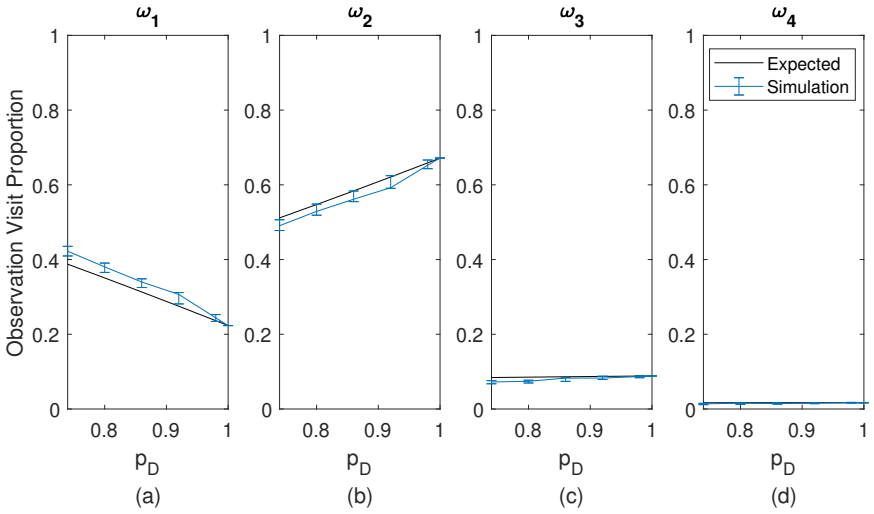

Figure B.9. Observation visit trends for case U_2 The solid black line indicates predicted performance based on Equation 6.2, while the blue line represents the mean across simulated operation of the proposed system, with the 0.25 and 0.75 quantile indicated by the error bars.

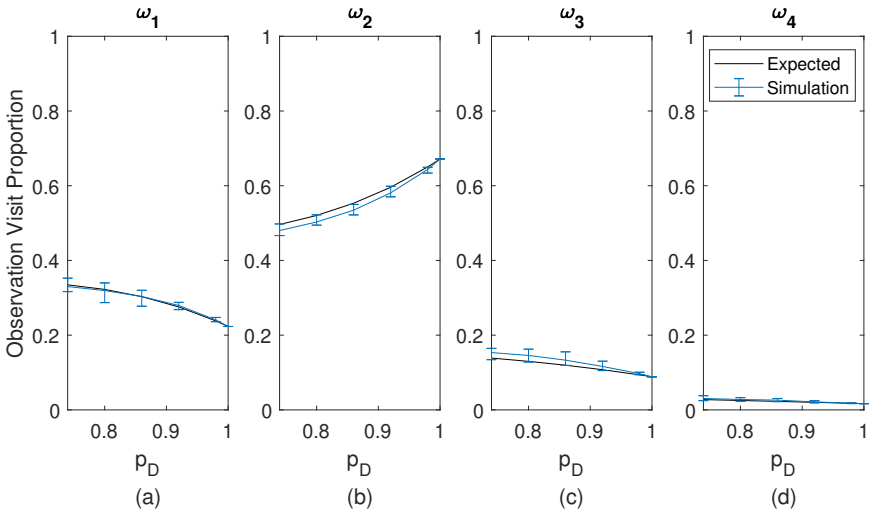

Figure B.10. Observation visit trends for case U_3 The solid black line indicates predicted performance based on Equation 6.2, while the blue line represents the mean across simulated operation of the proposed system, with the 0.25 and 0.75 quantile indicated by the error bars.

APPENDIX C:
Simulation Mechanics

To validate the proposed system, we conducted simulations using the procedures and apparatus described in this appendix. Each simulated attack starts from initialization conditions $b(0) = [1, 0, 0, 0, 0]$ and $s(0) = s_1$. From there, the workflow is as follows:

- Proposed system
  Receives $\omega(t)$ from simulation controller
  Performs belief update
  Updates POMDPX specification file
  Conducts DESPOT planning resulting in $a(t)$
  Feeds $a(t)$ to simulation controller
- Simulation controller
  Receives $a(t)$ from the proposed system
  Returns $\omega(t)$ based on hidden $s(t)$, POMDP specification, and pseudo-random number generator

Simulated operation continues until $s(t) = s_5$; at which point, the simulation controller stops the simulation event.

C.1 Hardware

Simulations of different uncertainty conditions U_i and probabilities of detection p_D were conducted in parallel across eight Dell PowerEdge R420 blade servers, each having between 24 and 48 central processing units (CPU). CPUs were Intel Xeon E5-2430. Each server had enough random access memory (RAM) installed to ensure that between four and eight GB RAM was available per CPU. Simulations were manually allocated to each server to ensure that a ratio of at least one CPU per active DESPOT process was maintained.

C.2 Software

Software versions and code snippets are listed to facilitate reproducing our results. Simulation and analysis required software as follows:

- Pre-/Post-processing and analysis
  - MATLAB 2019b with MDP Toolbox [81]
  - pomdp-solve [82]
  - Python 3.6.9 with numpy 1.18.2 and matplotlib 2.1.1
- Proposed system
  - Python 3.6.9 with numpy 1.18.2, subprocess, and re packages
  - DESPOT [75]
- Simulation control
  - Python 3.6.9 with numpy 1.18.2 and random packages
  - GNU bash 4.4.20

Critical functions developed to support simulation of the proposed system are included here. The simulation controller employs the pseudo-random number generators in Python 3.6.9 to return observation $\omega(1)$ and state $s(1)$ based on input state $s(0)$, action $a(0)$, and the POMDP specification of P and O as follows:

```
def simulation_control(state_0, action_0, P, O):
        Pm = numpy.array(P[action_0][state_0])
        Pm = numpy.cumsum(Pm)
        state_1 = next(x for x, val in enumerate(Pm) if val > random.
            random())
        Om = numpy.array(O[state_1])
        Om = numpy.cumsum(Om)
        obs_1 = next(x for x, val in enumerate(Om) if val > random.
            random())
        return (obs_1, state_1)
```

The Bayesian update takes previous belief state $b(0)$, observation $\omega(0)$, action $a(0)$, and returns new belief state $b(1)$ based on the POMDP formulation of P and O. The function is performed as follows:

```
def bayes_belief_up(b_0, obs_0, a_0, P, O):
```

```
        n = len(P[0])
        temp = numpy.zeros(n)
        b_1 = numpy.zeros(n)
        for k in range(0,n):
               for j in range(0,n):
                      temp[j]=P[a_0][j][k]*b_0[j]
               b_1[k]= O[k][obs_0]*sum(temp)
        m = sum(b_1)
        for k in range(0,n):
               b_1[k] = b_1[k]/m
        return b_1
```

The simulations are controlled via Python 3.6.9 using a subprocess to leverage DESPOT software [75] to determine the next optimal action based on belief state b_0, assuming that the POMDPX is stored at `file_loc`, the DESPOT executable is located at `ex_loc`, and DESPOT output is written to `action_loc`. Parameters selected as described in Chapter 5 were hard coded via the options `-d`, `-t`, `-n` representing depth, computational time, and number of particles respectively. Code is as follows:

```
def DESPOT_action(b_0,file_loc,ex_loc,action_loc):
        POMDPX_update(b_0,file_loc)
        string = [ex_loc,'-s 1','--runs 1','-d 11','-t 40','-n 80','-
           m',file_loc]
        with open(action_loc,'w') as fout:
               subprocess.call(string,stdout = fout)
        return Get_action(action_loc)
```

Function `POMDPX_update` updates the POMDPX specification file following the format in Appendix D with the new belief state b_0 as follows:

```
def POMDPX_update(b_0,file_loc):
        new_line = "                <ProbTable>"
        n = len(b_0)
        for k in range(0,n-1):
               new_line += str(b_0[k])
               new_line += ' '
```

```
        new_line += str(b_0[n-1])
        new_line += "</ProbTable>\n"
        with open(file,'r+') as f:
               text = f.readlines()
               text[22] = new_line
               f.close()
        with open(file,'w') as t:
               t.writelines(text)
               t.close()
        return 0
```

Function `Get_action` recovers $a(1)$ from the text output of DESPOT using the regular expressions package `re` as follows:

```
def Get_action(action_loc):
        with open(action_loc, 'r') as f:
               data = f.read()
               action = re.search(r"- Action = (\d):", data);
               f.close()
        return int(action.group(1))
```

APPENDIX D:
Sample Model Specification File

This appendix includes an example POMDP specification for use in the proposed system. The specification is detailed in extensible markup language based on the template provided by [75]. This example is written for the five state, three action model described in Chapter 7 and summarized in Appendix A. This example file is set up for uncertainty condition U_3, $p_D = 0.9$ and $b(t) = [0.0, 0.0, 0.309, 0.691, 0.0]$. The code snippets included in Appendix C are written expecting formatting exactly as listed here.

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<pomdpx version='0.1' id='autogenerated' xmlns:xsi='http://www.w3.org/2001/
    XMLSchema-instance' xsi:noNamespaceSchemaLocation="pomdpx.xsd">
    <Discount>0.95</Discount>
    <Variable>
        <StateVar vnamePrev="attack_0" vnameCurr="attack_1" fullyObs="false
            ">
            <ValueEnum>st ts vs el xx</ValueEnum>
        </StateVar>
        <ObsVar vname="IDS">
            <ValueEnum>st ts vs el xx</ValueEnum>
        </ObsVar>
        <ActionVar vname="defense">
            <ValueEnum>nl d1 d2</ValueEnum>
        </ActionVar>
        <RewardVar vname="reward"/>
    </Variable>
    <InitialStateBelief>
        <CondProb>
            <Var>attack_0</Var>
            <Parent>null</Parent>
            <Parameter type="TBL">
                <Entry>
                    <Instance>-</Instance>
```

```
            <ProbTable>0.0 0.0 0.309 0.691 0.0</ProbTable>
        </Entry>
    </Parameter>
</CondProb>
</InitialStateBelief>
<RewardFunction>
    <Func>
        <Var>reward</Var>
        <Parameter type="TBL">
            <Entry>
                <Instance>nl -</Instance>
                <ValueTable>0 0 0 0 -959</ValueTable>
            </Entry>
            <Entry>
                <Instance>d1 -</Instance>
                <ValueTable>-0.635 -0.635 -0.635 -0.635 -959.635</
                    ValueTable>
            </Entry>
            <Entry>
                <Instance>d2 -</Instance>
                <ValueTable>-9.59 -9.59 -9.59 -9.59 -968.59</ValueTable>
            </Entry>
        </Parameter>
    </Func>
</RewardFunction>
<ObsFunction>
    <CondProb>
        <Var>IDS</Var>
        <Parameter type="TBL">
            <Entry>
                <Instance>st -</Instance>
                <ProbTable>0.90 0.10 0.00 0.00 0.00</ProbTable>
            </Entry>
```

```
            <Entry>
                <Instance>ts -</Instance>
                <ProbTable>0.05 0.90 0.05 0.00 0.00</ProbTable>
            </Entry>
            <Entry>
                <Instance>vs -</Instance>
                <ProbTable>0.00 0.05 0.90 0.05 0.00</ProbTable>
            </Entry>
            <Entry>
                <Instance>el -</Instance>
                <ProbTable>0.00 0.00 0.05 0.90 0.05</ProbTable>
            </Entry>
            <Entry>
                <Instance>xx -</Instance>
                <ProbTable>0.00 0.00 0.00 0.10 0.90</ProbTable>
            </Entry>
        </Parameter>
    </CondProb>
</ObsFunction>
<StateTransitionFunction>
    <CondProb>
        <Var>attack_1</Var>
        <Parent>defense attack_0</Parent>
        <Parameter type="TBL">
            <Entry>
                <Instance>nl st -</Instance>
                <ProbTable>0.6611 0.23 0.0856 0.0233 0.0</ProbTable>
            </Entry>
            <Entry>
                <Instance>nl ts -</Instance>
                <ProbTable>0.0 0.9235 0.0687 0.0078 0.0</ProbTable>
            </Entry>
            <Entry>
                <Instance>nl vs -</Instance>
                <ProbTable>0.0 0.0 0.7890 0.2110 0.0</ProbTable>
```

```
</Entry>
<Entry>
   <Instance>nl el -</Instance>
   <ProbTable>0.0 0.0 0.0 0.5 0.5</ProbTable>
</Entry>
<Entry>
   <Instance>* xx -</Instance>
   <ProbTable>0.0 0.0 0.0 0.0 1.0</ProbTable>
</Entry>
<Entry>
   <Instance>d1 st -</Instance>
   <ProbTable>0.88703 0.07667 0.02853 0.00777 0.0</ProbTable>
</Entry>
<Entry>
   <Instance>d1 ts -</Instance>
   <ProbTable>0.6667 0.3078 0.0229 0.0026 0.0</ProbTable>
</Entry>
<Entry>
   <Instance>d1 vs -</Instance>
   <ProbTable>0.6667 0.0 0.2633 0.070 0.0</ProbTable>
</Entry>
<Entry>
   <Instance>d1 el -</Instance>
   <ProbTable>0.6666 0.0 0.0 0.1667 0.1667</ProbTable>
</Entry>
<Entry>
   <Instance>d2 st -</Instance>
   <ProbTable>0.9987 0.0009 0.0003 0.0001 0.0</ProbTable>
</Entry>
<Entry>
   <Instance>d2 ts -</Instance>
   <ProbTable>0.99609375 0.00360742 0.00026836 0.000030469
      0.0</ProbTable>
</Entry>
<Entry>
```

```
                <Instance>d2 vs -</Instance>
                <ProbTable>0.99609 0.0 0.00309 0.00082031 0.0</ProbTable>
            </Entry>
            <Entry>
                <Instance>d2 el -</Instance>
                <ProbTable>0.99609375 0.0 0.0 0.00195313 0.00195313</
                    ProbTable>
            </Entry>
        </Parameter>
    </CondProb>
</StateTransitionFunction>
</pomdpx>
```

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

[1] K. Townsend. (2019, Mar. 21). Global security spend set to grow to $133.8 billion by 2022: IDC. *Security Week*. [Online]. Available: https://www.securityweek.com/global-security-spend-set-grow-1338-billion-2022-idc

[2] IBM Security. (2019). Cost of a data breach report. *IBM Security*. [Online]. Available: https://www.ibm.com/downloads/cas/ZBZLY7KL?_ga=2.236878469.543068773.1585849973-1023088204.1585849973

[3] A. M. Ortiz, "Data breaches: Range of consumer risks highlights limitations of identity theft services," Washington, DC, USA, GAO Report No. GAO-19-230, 2019.

[4] Symantec. (2019). Internet security threat report. *Symantec*. [Online]. Available: https://docs.broadcom.com/doc/istr-24-2019-en

[5] U.S. Cyber Security and Information Assurance Interagency Working Group Networking and Information Technology Research and Development Subcommittee. (2010). Cyber Security and Information Assurance Interagency Working Group Cybersecurity Research and Development Recommendations. [Online]. Available: https://www.nitrd.gov/Publications/PublicationDetail.aspx?pubid=24

[6] B. Ward, S. Gomez, R. W. Skowyra, D. Bigelow, J. Martin, J. Landry, and H. Okhravi, "Survey of cyber moving targets," MIT Lincoln Laboratory, Lexington, Massachusetts, Tech. Rep. 1228, January 2018.

[7] M. Carvalho and R. Ford, "Moving-target defenses for computer networks," *IEEE Security & Privacy*, vol. 12, no. 2, pp. 73–76, Mar 2014.

[8] W. Connell, L. H. Pham, and S. Philip, "Analysis of concurrent moving target defenses," in *Proceedings of the 5th ACM Workshop on Moving Target Defense*. New York, NY, USA: ACM, 2018, pp. 21–30.

[9] W. Connell, D. A. Menasce, and M. Albanese, "Performance modeling of moving target defenses with reconfiguration limits," *IEEE Trans. on Dependable and Secure Computing*, pp. 1–1, 2018.

[10] S. A. DeLoach, X. Ou, R. Zhuang, and S. Zhang, "Model-driven, moving-target defense for enterprise network security," in *Models@run.time: Foundations, Applications, and Roadmaps*, N. Bencomo, R. France, B. H. C. Cheng, and U. Aßmann, Eds. Cham: Springer International Publishing, 2014, pp. 137–161.

[11] J. M. Benyus, *Biomimicry: Innovation inspired by nature*. New York, NY, USA: Perennial, 2002.

[12] W. Mazurczyk, S. Drobniak, and S. Moore, "Towards a systematic view on cybersecurity ecology," *CoRR*, vol. abs/1505.04207, 2015. Available: http://arxiv.org/abs/1505.04207

[13] H. M. ter Hofstede and J. M. Ratcliffe, "Evolutionary escalation: The bat–moth arms race," *Journal of Experimental Biology*, vol. 219, no. 11, pp. 1589–1602, 2016. Available: http://jeb.biologists.org/content/219/11/1589

[14] M. B. Fenton and J. H. Fullard, "Moth hearing and the feeding strategies of bats: Variations in the hunting and echolocation behavior of bats may reflect a response to hearing-based defenses evolved by their insect prey," *American Scientist*, vol. 69, no. 3, pp. 266–275, 1981. Available: http://www.jstor.org/stable/27850425

[15] T. Myers, "Unidentified moth," May 2020, unpublished.

[16] E. M. Hutchins, M. J. Cloppert, and R. M. Amin, "Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains," *Leading Issues in Information Warfare & Security Research*, vol. 1, no. 1, p. 80, 2011.

[17] S. Panjwani, S. Tan, K. M. Jarrin, and M. Cukier, "An experimental evaluation to determine if port scans are precursors to an attack," in *2005 International Conference on Dependable Systems and Networks (DSN'05)*, June 2005, pp. 602–611.

[18] L. Spitzner, *Honeypots: Tracking hackers*. Boston, MA, USA: Addison-Wesley Reading, 2003.

[19] Z. Zhan, M. Xu, and S. Xu, "Characterizing honeypot-captured cyber attacks: Statistical framework and case study," *IEEE Trans. on Information Forensics and Security*, vol. 8, no. 11, pp. 1775–1789, 2013.

[20] J. G. Kemeny and J. L. Snell, *Finite Markov chains*. New York: Van Nostrand, 1959.

[21] M. L. Puterman, *Markov decision processes*. New York: John Wiley and Sons, Inc., 1994.

[22] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[23] C. C. White and W. T. Scherer, "Solution procedures for partially observed Markov decision processes," *Operations Research*, vol. 37, no. 5, pp. 791–797, 1989. Available: http://www.jstor.org/stable/171024

[24] A. Somani, N. Ye, D. Hsu, and W. Lee, "DESPOT: Online POMDP planning with regularization," in *Advances in neural information processing systems*, 2013, pp. 1772–1780.

[25] S. Ross, J. Pineau, S. Paquet, and B. Chaib-Draa, "Online planning algorithms for POMDPs," *Journal of Artificial Intelligence Research*, vol. 32, pp. 663–704, 2008.

[26] D. Silver and J. Veness, "Monte-Carlo planning in large POMDPs," in *Advances in neural information processing systems*, 2010, pp. 2164–2172.

[27] O. P. Kreidl and T. M. Frazier, "Feedback control applied to survivability: a host-based autonomic defense system," *IEEE Trans. on Reliability*, vol. 53, no. 1, pp. 148–166, March 2004.

[28] C. Sarraute, O. Buffet, and J. Hoffmann, "POMDPs make better hackers: Accounting for uncertainty in penetration testing," in *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[29] R. Tipireddy, S. Chatterjee, P. Paulson, M. Oster, and M. Halappanavar, "Agent-centric approach for cybersecurity decision-support with partial observability," in *2017 IEEE International Symposium on Technologies for Homeland Security (HST)*, April 2017, pp. 1–6.

[30] S. A. Zonouz, H. Khurana, W. H. Sanders, and T. M. Yardley, "RRE: A game-theoretic intrusion response and recovery engine," *IEEE Trans. on Parallel and Distributed Systems*, vol. 25, no. 2, pp. 395–406, Feb 2014.

[31] E. Miehling, M. Rasouli, and D. Teneketzis, "A POMDP approach to the dynamic defense of large-scale cyber networks," *IEEE Trans. on Information Forensics and Security*, vol. 13, no. 10, pp. 2490–2505, Oct 2018.

[32] S. Musman, L. Booker, A. Applebaum, and B. Edmonds, "Steps toward a principled approach to automating cyber responses," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, T. Pham, Ed., International Society for Optics and Photonics. SPIE, 2019, vol. 11006, pp. 490 – 504.

[33] H. Okhravi, T. Hobson, D. Bigelow, and W. Streilein, "Finding focus in the blur of moving-target techniques," *IEEE Security Privacy*, vol. 12, no. 2, pp. 16–26, Mar 2014.

[34] S. Abraham and S. Nair, "Cyber security analytics: a stochastic model for security quantification using absorbing Markov chains," *Journal of Communications*, vol. 9, no. 12, pp. 899–907, 2014.

[35] C. Wang and V. M. Bier, "Quantifying adversary capabilities to inform defensive resource allocation," *Risk Analysis*, vol. 36, no. 4, pp. 756–775, 2016.

[36] W. J. Connell, "A quantitative framework for cyber moving target defenses," Ph.D. dissertation, Dept. of Info. Sci. and Tech., George Mason Univ., 2017. Available: http://ebot.gmu.edu/handle/1920/11325

[37] Lockheed Martin. Cyber Kill Chain. *Lockheed Martin*. [Online]. Available: https://lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html

[38] H. Okhravi, W. W. Streilein, and K. S. Bauer, "Moving target techniques: leveraging uncertainty for cyber defense," *Licoln Laboratory Journal*, 2016.

[39] E. Goldberg. (2017, September). New Email Security Report from IRONSCALES Identifies Email Phishing Attack Detection, Mitigation and Remediation as Biggest Challenge for Security Teams. *PRWeb*. [Online]. Available: http://www.prweb.com/releases/2017/09/prweb14742215.htm. Online.

[40] D. Ariu, E. Frumento, and G. Fumera, "Social engineering 2.0: A foundational work," in *Proceedings of the Computing Frontiers Conference* (CF'17). New York, NY, USA: ACM, 2017, pp. 319–325. Available: http://doi.acm.org/10.1145/3075564.3076260

[41] G. Rößling and M. Müller, "Social engineering: a serious underestimated problem," *SIGCSE Bull.*, vol. 41, no. 3, pp. 384–384, July 2009. Available: http://doi.acm.org/10.1145/1595496.1563026

[42] G. F. Lyon, *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. Insecure, 2009.

[43] J. Treurniet, "A network activity classification schema and its application to scan detection," *IEEE/ACM Trans. on Networking*, vol. 19, no. 5, pp. 1396–1404, Oct 2011.

[44] C. Leckie and R. Kotagiri, "A probabilistic approach to detecting network scans," in *NOMS 2002. IEEE/IFIP Network Operations and Management Symposium. ' Management Solutions for the New Communications World'(Cat. No.02CH37327)*, 2002, pp. 359–372.

[45] K. Kuczma. (2018). Microsoft Targeted by 8 of 10 Top Vulnerabilities in 2018. *Recorded Future*. [Online]. Available: https://www.recordedfuture.com/top-vulnerabilities-2018/

[46] B. Coppens, B. De Sutter, and K. De Bosschere, "Protecting your software updates," *IEEE Security Privacy*, vol. 11, no. 2, pp. 47–54, March 2013.

[47] B. Brown, D. Ennis, J. Kaplan, and J. Rosenthal, "To survive in the age of advanced cyberthreats, use 'active defense'," *Digital McKinsey*, November 2017. Available: https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/to-survive-in-the-age-of-advanced-cyberthreats-use-active-defense

[48] D. Liu, Z. Li, K. Du, H. Wang, B. Liu, and H. Duan, "Don't let one rotten apple spoil the whole barrel: towards automated detection of shadowed domains," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (CCS '17). New York, NY, USA: ACM, 2017, pp. 537–552. Available: http://doi.acm.org/10.1145/3133956.3134049

[49] N. Nissim, R. Yahalom, and Y. Elovici, "USB-based attacks," *Computers & Security*, vol. 70, pp. 675 – 688, 2017. Available: http://www.sciencedirect.com/science/article/pii/S0167404817301578

[50] M. Tischer, Z. Durumeric, S. Foster, S. Duan, A. Mori, E. Bursztein, and M. Bailey, "Users really do plug in USB drives they find," in *2016 IEEE Symposium on Security and Privacy*, 2016, pp. 306–319.

[51] H. Debar, M. Dacier, and A. Wespi, "Towards a taxonomy of intrusion-detection systems," *Computer Networks*, vol. 31, no. 8, pp. 805 – 822, 1999.

[52] P. Nespoli, D. Papamartzivanos, F. G. Mármol, and G. Kambourakis, "Optimal countermeasures selection against cyber attacks: A comprehensive survey on reaction frameworks," *IEEE Communications Surveys Tutorials*, vol. 20, no. 2, pp. 1361–1396, Secondquarter 2018.

[53] D. Ariu, G. Giacinto, and R. Perdisci, "Sensing attacks in computers networks with hidden Markov models," in *International Workshop on Machine Learning and Data Mining in Pattern Recognition*. Springer, 2007, pp. 449–463.

[54] N. Ye, Y. Zhang, and C. Borror, "Robustness of the Markov-chain model for cyber-attack detection," *IEEE Trans. on Reliability*, vol. 53, no. 1, pp. 116–123, March 2004.

[55] G. Brogi and E. Di Bernardino, "Hidden Markov models for advanced persistent threats," 2017, unpublished. Available: https://hal.archives-ouvertes.fr/hal-01549196/

[56] A. Bar, B. Shapira, L. Rokach, and M. Unger, "Identifying attack propagation patterns in honeypots using Markov chains modeling and complex networks analysis," in *2016 IEEE International Conference on Software Science, Technology and Engineering (SWSTE)*, June 2016, pp. 28–36.

[57] R. Gore, J. Padilla, and S. Diallo, "Markov chain modeling of cyber threats," *The Journal of Defense Modeling and Simulation*, vol. 14, no. 3, pp. 233–244, 2017.

[58] A. S. Sendi, M. Dagenais, M. Jabbarifar, and M. Couture, "Real time intrusion prediction based on optimized alerts with hidden Markov model," *JNW*, vol. 7, pp. 311–321, 2012.

[59] A. Årnes, F. Valeur, G. Vigna, and R. A. Kemmerer, "Using hidden Markov models to evaluate the risks of intrusions," in *Recent Advances in Intrusion Detection*, D. Zamboni and C. Kruegel, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 145–164.

[60] X. Li, "Network security risk assessment method based on the improved hidden Markov model." *International Journal of Simulation–Systems, Science & Technology*, vol. 17, no. 36, 2016.

[61] S. M. Rajasooriya, C. P. Tsokos, and P. K. Kaluarachchi, "Cyber security: Nonlinear stochastic models for predicting the exploitability," *Journal of information Security*, vol. 8, no. 02, p. 125, 2017.

[62] S. Abraham and S. Nair, "A novel architecture for predictive cybersecurity using non-homogenous Markov models," in *2015 IEEE Trustcom/BigDataSE/ISPA*, Aug 2015, vol. 1, pp. 774–781.

[63] S. Abraham and S. Nair, "Predictive cyber security analytics framework: A non-homogenous Markov model for security quantification," in *Second International Conference of Security, Privacy and Trust Management (SPTM)*, 2014.

[64] P. K. Kaluarachchi, "Cybersecurity: Stochastic analysis and modelling of vulnerabilities to determine the network security and attackers behavior," Ph.D. dissertation, Dept. of Math. and Stat., Univ. of South Florida, 2017. Available: https://scholarcommons.usf.edu/cgi/viewcontent.cgi?referer=https://www.google.com/&httpsredir=1&article=8059&context=etd

[65] M. Husák, J. Komárková, E. Bou-Harb, and P. Celeda, "Survey of attack projection, prediction, and forecasting in cyber security," *IEEE Communications Surveys & Tutorials*, pp. 640–660, 2018.

[66] J. H. H. Jafarian, E. Al-Shaer, and Q. Duan, "Spatio-temporal address mutation for proactive cyber agility against sophisticated attackers," in *Proceedings of the First ACM Workshop on Moving Target Defense* (MTD '14). New York, NY, USA: ACM, 2014, pp. 69–78.

[67] Q. Duan, E. Al-Shaer, M. Islam, and H. Jafarian, "CONCEAL: A strategy composition for resilient cyber deception-framework, metrics and deployment," in *2018 IEEE Conference on Communications and Network Security (CNoS)*, 2018, pp. 1–9.

[68] B. Burshteyn, "Security via dynamic data movement in a cloud-based environment," U.S. Patent US20 180 217 997A1, Jul, 2018.

[69] A. J. O'Donnell and H. Sethu, "On achieving software diversity for improved network security using distributed coloring algorithms," in *Proceedings of the 11th ACM Conference on Computer and Communications Security* (CCS '04). New York, NY, USA: ACM, 2004, pp. 121–131.

[70] M. Azab, B. M. Mokhtar, A. S. Abed, and M. Eltoweissy, "Smart moving target defense for linux container resiliency," in *2016 IEEE 2nd International Conference on Collaboration and Internet Computing (CIC)*, Nov 2016, pp. 122–130.

[71] M. J. Kochenderfer, *Decision making under uncertainty: theory and application*. Cambridge, Massachusetts: MIT Press, 2015.

[72] E. Miehling, M. Rasouli, and D. Teneketzis, "Optimal defense policies for partially observable spreading processes on Bayesian attack graphs," in *Proceedings of the Second ACM Workshop on Moving Target Defense* (MTD '15). New York, NY, USA: ACM, 2015, pp. 67–76.

[73] A. Cassandra, M. L. Littman, and N. L. Zhang, "Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes," in *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1997, pp. 54–61.

[74] T. Jaakkola, S. P. Singh, and M. I. Jordan, "Reinforcement learning algorithm for partially observable markov decision problems," in *Advances in neural information processing systems*, 1995, pp. 345–352.

[75] N. Ye, A. Somani, D. Hsu, and W. Lee. Approximate POMDP Planning Online Toolkit. National University of Singapore. [Online]. Available: https://github.com/AdaCompNUS/despot

[76] N. Kalanjee. (2016, Jul). 54 days to recovery: The impact of cyber crime. Hewlett Packard. [Online]. Available: https://blog.ext.hp.com/t5/BusinessBlog-en/54-days-to-recovery-The-impact-of-cyber-crime/ba-p/6720

[77] Mandiant Intelligence Center. (2013, April). APT1: Exposing one of China's cyber espionage units. [Online]. Available: https://www.fireeye.com/content/dam/fireeye-www/services/pdfs/mandiant-apt1-report.pdf

[78] Verizon. (2019, May). 2019 data breach investigations report. [Online]. Available: https://enterprise.verizon.com/resources/reports/2019-data-breach-investigations-report.pdf

[79] "Geometric series," 2008. Available: http://search.credoreference.com/content/entry/penguinmath/geometric_series/0

[80] A. McAbee, M. Tummala, and J. McEachen. Poster: A moving target defense scheme with overhead control. 41st IEEE Symposium on Security and Privacy. [Online]. Available: https://www.ieee-security.org/TC/SP2020/program-posters.html

[81] M. J. Cros. Markov Decision Processes Toolbox. MATLAB Central File Exchange. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/25786-markov-decision-processes-mdp-toolbox

[82] A. R. Cassandra. (2020). The POMDP page. [Online]. Available: http://www.pomdp.org/

[83] S. Ross, J. Pineau, B. Chaib-draa, and P. Kreitmann, "A Bayesian approach for learning and planning in partially observable Markov decision processes," *Journal of Machine Learning Research*, vol. 12, no. May, pp. 1729–1770, 2011.

[84] K. Azizzadenesheli, A. Lazaric, and A. Anandkumar, "Reinforcement learning of POMDPs using spectral methods," *CoRR*, vol. abs/1602.07764, 2016. Available: http://arxiv.org/abs/1602.07764

[85] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.

[86] D. B. Thomas, J. A. Bower, and W. Luk, "Hardware architectures for Monte-Carlo based financial simulations," in *2006 IEEE International Conference on Field Programmable Technology*, 2006, pp. 377–380.

[87] S. Ayubian, S. Alawneh, M. Richard, and J. Thijssen, "Implementation and performance of a GPU-based Monte-Carlo framework for determining design ice load," in *2017 International Conference on High Performance Computing Simulation (HPCS)*, 2017, pp. 109–116.

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California