NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

OPTIMIZATION OF MOVING TARGET DEFENSE USING A PARTIALLY OBSERVABLE MARKOV DECISION PROCESS AND DETERMINIZED SPARSE PARTIALLY OBSERVABLE TREE

by

Kelsey M. Shevock

June 2020

Co-Advisors: John C. McEachen
Murali Tummala

THIS PAGE INTENTIONALLY LEFT BLANK

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

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** June 2020	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE
OPTIMIZATION OF MOVING TARGET DEFENSE USING A PARTIALLY OBSERVABLE MARKOV DECISION PROCESS AND DETERMINIZED SPARSE PARTIALLY OBSERVABLE TREE

5. FUNDING NUMBERS

6. AUTHOR(S) Kelsey M. Shevock

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)

Those who defend systems against cyber-attacks can use moving target defense (MTD) to their advantage. However, optimal MTD techniques have yet to be sufficiently explored. In terms of cost-benefit analysis, the desired level of attack suppression will come at the cost of network availability, and optimization tools might be able to harness the advantages of MTD without undue sacrifice. This thesis formulates an attack/defense scenario as a partially observable Markov decision process (POMDP) to facilitate optimal MTD of a host. We develop a system in which service and IP reconfigurations can be employed as defense against a five-stage attack to maximize system availability and minimize cost. With an attack/defense scenario involving five attack stages and two defense options, we explore the utility of the Determinized Sparse Partially Observable Tree (DESPOT) algorithm for online optimal defense selection using the POMDP formulation. We compare optimization of the system for three different cases of the POMDP with varying levels of uncertainty (i.e., probability of detection) representing potential real-world scenarios. A significant result of this thesis is our development of a framework for optimizing MTD techniques. We also demonstrate, within the limitations of this research, how to determine the bounds for best performance when using DESPOT as an MTD controller.

14. SUBJECT TERMS
moving target defense, MTD, partially observable Markov decision process, POMDP, Determinized Sparse Partially Observable Tree, DESPOT, service reconfiguration, IP reconfiguration, attacker, defender, optimization

15. NUMBER OF PAGES
93

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.

OPTIMIZATION OF MOVING TARGET DEFENSE USING A PARTIALLY OBSERVABLE MARKOV DECISION PROCESS AND DETERMINIZED SPARSE PARTIALLY OBSERVABLE TREE

Kelsey M. Shevock
Lieutenant, United States Navy
BS, U.S. Naval Academy, 2014

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2020

Approved by: John C. McEachen
Co-Advisor

Murali Tummala
Co-Advisor

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

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Those who defend systems against cyber-attacks can use moving target defense (MTD) to their advantage. However, optimal MTD techniques have yet to be sufficiently explored. In terms of cost-benefit analysis, the desired level of attack suppression will come at the cost of network availability, and optimization tools might be able to harness the advantages of MTD without undue sacrifice. This thesis formulates an attack/defense scenario as a partially observable Markov decision process (POMDP) to facilitate optimal MTD of a host. We develop a system in which service and IP reconfigurations can be employed as defense against a five-stage attack to maximize system availability and minimize cost. With an attack/defense scenario involving five attack stages and two defense options, we explore the utility of the Determinized Sparse Partially Observable Tree (DESPOT) algorithm for online optimal defense selection using the POMDP formulation. We compare optimization of the system for three different cases of the POMDP with varying levels of uncertainty (i.e., probability of detection) representing potential real-world scenarios. A significant result of this thesis is our development of a framework for optimizing MTD techniques. We also demonstrate, within the limitations of this research, how to determine the bounds for best performance when using DESPOT as an MTD controller.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

DESPOT	Determinized Sparse Partially Observable Tree
EL	exploit launch
IR	IP reconfiguration
MDP	Markov decision process
MTD	moving target defense
N	no action
POMDP	partially observable Markov decision process
S	start
SR	service reconfiguration
TS	target scan
VS	vulnerability scan
X	attack

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Cyber-attacks continue to be a major threat as well as the fastest growing crime in society today [1]. The use of computer systems, electronic data, and electronic data transfer is vital to federal agencies in carrying out their work and missions [2]. The protection of these systems, data, and communication methods is therefore vital to the protection of sensitive information and the protection of the nation.

Both traditional network systems and traditional network defenses have similar characteristics of being static in nature; thus, as an adversary gains information on a target over time, they are more likely to succeed in launching a successful attack [3]. Moving target defense was introduced in 2009 as a method to deviate from this static environment towards a more dynamic and unpredictable environment, meaning that time would no longer be an advantage to the adversary [3]. There has been a significant amount of research in moving target defense in the last decade; however, there has been less research dedicated to its optimization in terms of cost/reward for the defender [4].

The motivation for this thesis lies in optimizing moving target defense. The research conducted in this thesis brings more clarity to harnessing the advantages of moving target defense to achieve desired attack suppression while minimizing undue sacrifice of various cost factors. This thesis directly contributes to improving network security.

A. OBJECTIVE

The objective of this thesis is to optimize moving target defense techniques by formulating an attack and defense model as a partially observable Markov decision process (POMDP) to facilitate such techniques. The attack model is derived from the intrusion kill chain, and the defense model involves two different moving target defense techniques, namely IP reconfigurations and service reconfigurations.

DESPOT (Determinized Sparse Partially Observable Tree) is an online planning tool used to solve the POMDP model in this thesis. The performance of DESPOT as an MTD controller in uncertain environments will also be reviewed in this thesis. The

performance is analyzed to determine how well decision making follows the optimal policy of the POMDP model.

B. RELATED WORK

The related work for this thesis can be divided into two categories: works that were critical in the development and solution of the POMDP model in this thesis and works that are related in topic.

The moving target defense techniques employed in this thesis, as well as their respective costs, are directly related to the work performed by Connell et al. [4]. They propose a concurrent moving target defense technique with both service and IP reconfigurations. The transition matrix of the POMDP model developed in this thesis is derived from the data collected by Panjwani et al. [5]. They conducted a study to determine the likelihood that port scans and vulnerability scans are precursors to attacks. This data allowed us to determine the transition probabilities for the attack model in our POMDP. Using these two references to build a cohesive model of attack defense dynamics was introduced in [6], which we used as the basis for the model in this thesis with transition probabilities discerned from analysis of honeypot data from [5].

The concept of the intrusion kill chain was used to develop a model on which to implement moving target defenses in this thesis. The intrusion kill chain is described in depth in [7]. The categories of moving target defense as well as the different techniques used to implement it are critical to this thesis and are catalogued in [3] and [8]. The concepts of Markov decision processes as well as POMDPs were critical in the development of the POMDP model in this thesis. Markov decision processes are covered in [9], and POMDPs are covered in [10]. Understanding POMDPs and approaches to finding their solutions in partially stochastic environments is thoroughly explored in [11] and [12]. DESPOT, which was the tool used to find the solution to our POMDP model, is thoroughly explained in [13].

C. ORGANIZATION

There are five chapters as well as three appendices in this thesis. Chapter II covers background information on the intrusion kill chain, moving target defense, Markov decision process including POMDPs, as well as POMDP solution methods and DESPOT. The methodology employed in this thesis in order to optimize the employment of moving target defense techniques is explored in Chapter III. Chapter IV details the implementation of the ideas described in the methodology as well as the results. The conclusions as well as considerations for future work are discussed in Chapter V. Appendix A contains the POMDP model developed in this thesis. Appendix B contains all of the MATLAB code used in the analysis of the POMDP models as well as the results obtained from DESPOT.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

The concepts essential to the proposed optimization scheme in this thesis are explored in this chapter. First, network intrusions and the intrusion kill chain are introduced. Next, traditional network intrusion defense methods are introduced with a focus on moving target detection. Then, possible modeling techniques, including POMDPs for optimizing moving target defense in an uncertain environment, are discussed. Finally, DESPOT is examined as a solution to POMDPs.

A. INTRUSION KILL CHAIN

Protected networks are always at risk to unauthorized intrusions, and as these intrusions continue to evolve in robustness and complexity, it is necessary to adapt network defense systems in a parallel fashion. One example of a threat that has gained notoriety in recent years for its ability to defeat traditional network defense systems is the advanced persistent threat (APT) [7]. APTs vary from traditional threats in that they are uniquely crafted to compromise a specific target and are designed to operate for a long period of time while evading detection [14]. In order to improve network defense system ability to protect against more complex and untraditional threats, such as APTs, Lockheed Martin developed the concept of the intrusion kill chain depicted in Figure 1 [7].

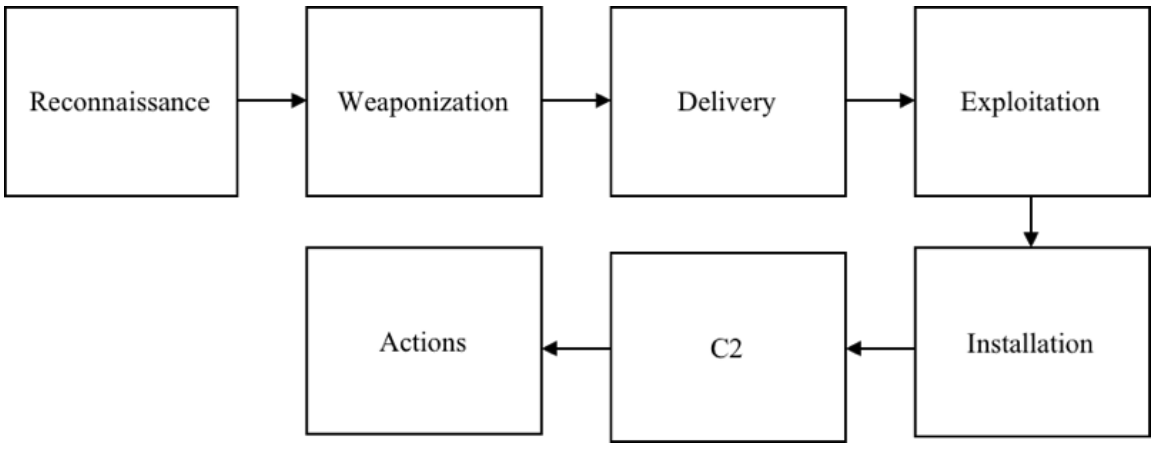

Figure 1. Intrusion Kill Chain. Adapted from [7].

The Intrusion Kill Chain is a process that describes the stages of complex network intrusions, such as APTs, from the adversary perspective [7]. The information gained in each stage is often times necessary in order to advance to the next stage of the process, and a disruption at any point in this chain is likely to prevent the process from proceeding forward [7]. There are seven stages: the reconnaissance stage is where data is gathered on the target, the weaponization stage is where the weapon or exploit is designed for the target, the delivery phase is where that weapon is transmitted to the target, the exploitation stage is where the weapon is triggered by the adversary and takes advantage of the target vulnerabilities, the installation stage is where the weapon establishes a way to maintain long term functionality within the target environment, the command and control stage is where the adversary establishes two way communication with the weapon for better control, and the action stage is where the weapon carries out the actions for which it was designed [7].

The Intrusion Kill Chain provides a model for the wide breadth of specialized complex intrusions used by APTs and categorizes the adversary goals, actions, and targeted vulnerabilities in such a way that creates a common framework across these intrusions [7]. This allows defenders to better identify their vulnerabilities and focus their defense capabilities against attacks that are specifically targeted for them [7]. With better focused defenses, defenders are able to adapt to an environment in which attacks are becoming more and more complex.

B. MOVING TARGET DEFENSE

Traditional network defenses are not always enough to successfully defend against increasingly complex network intrusions. Traditional network defenses focus on reducing system vulnerabilities and enhancing already existing protection measures, such as firewalls, but these defenses are still being overcome by sophisticated network intrusions [3]. Moving target defense was developed in 2009 as a new way of defending against network intrusions [3]. Traditional defense measures, even with constant improvements, do not change the static nature of network dynamics; however, the moving target defense changes the static nature to a dynamic one [3]. By changing the nature of network systems,

moving target defense increases the workload of the attacker and decreases the success rate of sophisticated network intrusions [3].

Moving target defense changes the network system from a static system to dynamic one by constantly altering the attack surface of the network which in turn makes it more difficult for an attacker to proceed along the intrusion kill chain [15]. Figure 2 shows that in a dynamic system, the attack surface is constantly moving around within the exploration space, which includes all the possible combinations of collected information from the attacker reconnaissance phase [3]. The attack surface of network system includes all the ways in which an attacker can access and connect to that system [15]. For example, when considering the intrusion kill chain stages, if the attacker has found a vulnerable service the target is using in the reconnaissance phase, and that service is randomly changed, the attacker will have to step back and regain that information before it can proceed to the next stage in the chain. When the attack surface is constantly changing, time is no longer an advantage to the attacker, which decreases the likelihood that complex, persistent threats, like the APT, will succeed.

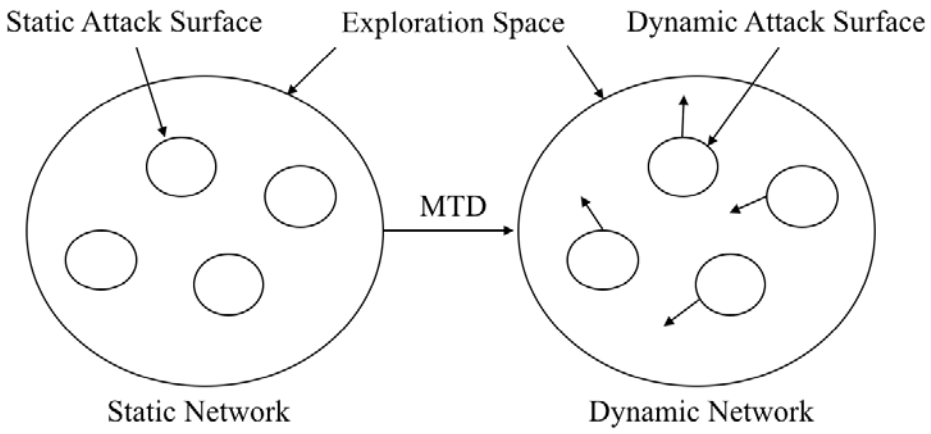

Figure 2. Static System to Dynamic System. Adapted from [3].

There are multiple different techniques that can be used to employ moving target defense, all of which change the attack surface of a system [15]. These MTD techniques fall into different categories that include dynamic data, software, runtime environment,

platforms, and networks; these are shown in Figure 3 [8]. Each of these techniques aims to protect different components of a system while also targeting the different phases of a cyber-attack [8]. This thesis focuses on techniques that fall into the category of dynamic platforms and networks. Dynamic platforms involve restricting system access of applications and restricting external connections to the system while dynamic networks involve obscuring network headers and obscuring the mapping of the network [8].

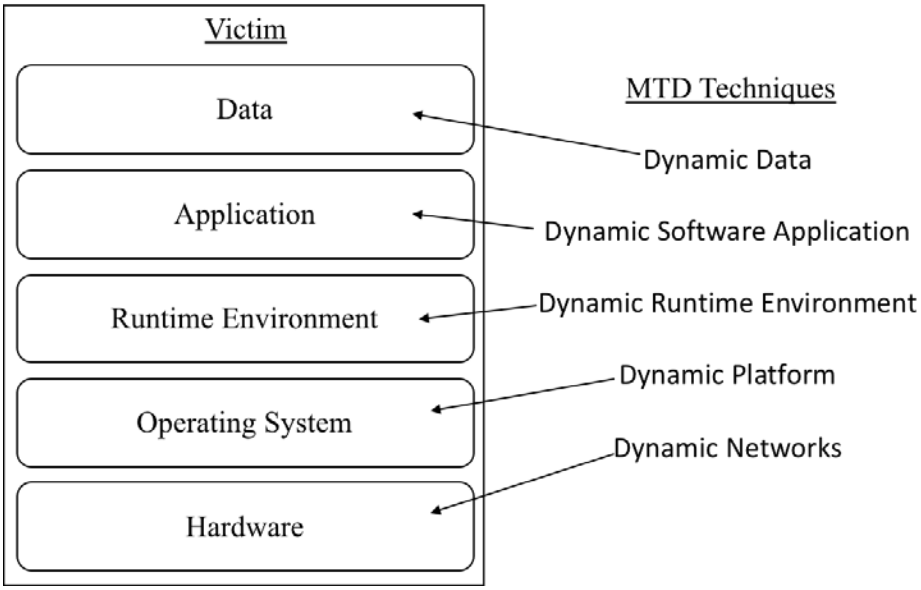

Figure 3. Moving Target Defense Techniques. Adapted from [8].

This thesis employs two types of moving target defense techniques, IP reconfigurations and service reconfigurations, in a concurrent manner in order to create a dynamic network and platform [4]. Internet Protocol (IP) reconfigurations are dynamic network techniques that randomize the IP address of the victim making it difficult for the attacker to locate the victim [4]. Service reconfigurations are dynamic platform techniques that randomize the services that the victim is using, such as a web service, making it difficult for the attacker to find a vulnerable service connected to the victim [4]. Different techniques are ideal against different types of attacks, and using multiple types of MTD techniques in one model increases the breadth of possible attacks that a system can defend against [4].

Unfortunately, most moving target defense techniques, including the two employed in this thesis, come with a cost to the defender in terms of equipment required for the techniques and temporary loss of network availability [4]. This thesis focuses on optimizing the employment of moving target defense techniques in terms of cost/reward to maximize the defender network availability while maintaining the desired level of defense against attacks.

C. MARKOV DECISION PROCESS

This thesis relies heavily on the concept of the Markov decision processes (MDP), which is a well-known model used to study optimization. A Markov decision process (MDP) is a mathematical model for a probabilistic system that allows one to plan for an optimal outcome when that outcome is only partially under the control of the decision maker [9]. This model can either represent a finite or infinite system in which a series of actions transition the system among a series of states, all with the goal of balancing immediate rewards/costs with future rewards/cost [9].

The components of an MDP include a set of decision epochs (T), a set of states ($s \in S$), a set of actions ($a \in A$), transition probabilities ($T(s, a, s') = p_t(s'|s, a)$), and rewards ($R(s, a, s') = r_t(s, a)$) where (s, a, s') defines a transition [11]. The collection of these components as a whole, $\{S, A, T, R\}$, is referred to as an MDP [11]. The following is a description of these components: decision epochs are points in time where decisions are made; system states are characterizations of the system at different points in time where the decision epochs occur; actions are the result of decision epochs that transition the system between states; transition probabilities are action dependent and represent the likelihood of transitioning to another state after an action has been taken; and rewards are the results of actions in the form of cost/reward to the system [9].

A Markov chain is a symbolic representation of an MDP, an example of which is depicted in Figure 4 with states and transition probabilities labeled [9]. There are two states represented by S_0 and S_2, and there are four transition probabilities represented by p_{00}, p_{01}, p_{11}, and p_{10}. If an action is taken in S_0, the model will either stay in S_0 or transition to S_1 according to the transition probabilities, p_{00} and p_{01}.

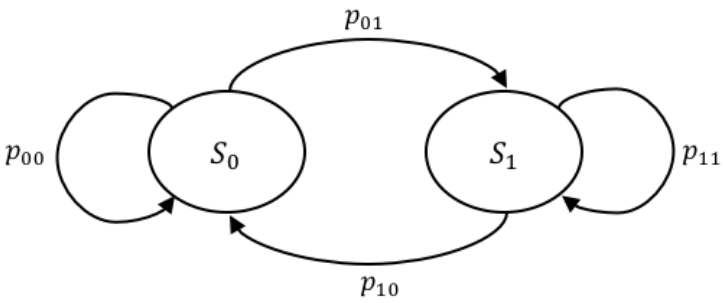

Figure 4. Markov Chain. Adapted from [9].

The transition probabilities of Markov chains can be represented as matrices, and the transition matrix for the Markov chain depicted in Figure 4 can be written as [9]:

$$P_T = \begin{bmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{bmatrix}. \tag{1}$$

Limiting state probabilities can be used to define the steady state of a transition matrix, and are often useful in analyzing MDPs [16]. The equation for limiting state probabilities can be written as [16]:

$$\overline{p} = \lim_{k\to\infty} p[k] = \lim_{k\to\infty} (P^k)^T p[0], \tag{2}$$

where $\overline{p}$ is the vector of steady-state probabilities and $p[0]$ is the initial state probability vector [16]. As $k \to \infty$, a transition matrix will reach a steady state where the rows become equal and the elements remain unchanged from that point forward [16].

In MDPs, agents make decisions based on policies ($\pi: S \to A$) which map states to actions [17]. The value of a policy is the long-term evaluation of a policy expressed as a numerical quantity. The value of a policy is defined by

$$V^{\pi}(s) = R\big(s, \pi(s)\big) + \sum_{s' \in S} T(s, \pi(s), s') \cdot \gamma \cdot V_{\pi}(s'), \tag{3}$$

which considers both immediate rewards and discounted rewards [17]. Discounted rewards are future rewards included in a policy that are discounted by the discount factor (γ) so that they impact decision making less than immediate rewards [17]. Optimal policies (π^*) are policies that have the highest values and maximize the expected utility if followed moving

forward [17]. The goal of an MDP is to maximize long term total discounted reward, which is accomplished by maximizing the value function

$$V^*(s) = \max_{a \in A} \left(R(s,a) + \sum_{s' \in S} T(s,a,s') \cdot \gamma \cdot V^*(s') \right), \quad (4)$$

known as the Bellman equation, which leads to the optimal policy [17]. There is only one V^* that solves the bellman equation; however, there may be multiple optimal policies that lead to the same V^* [17].

MDPs are Markovian and fully observable, meaning that decisions are made based only on the current state of the system and that those decisions are made with full certainty of the current state of the system [9]. During a cyber-attack, it is not always apparent to the system that it is being attacked or what stage of the attack is being carried out, thus it is not always apparent to a system what state of attack it is under. MDPs are not suitable to this type of scenario; however, they set the foundation for a model that is suitable, the POMDP.

D. PARTIALLY OBSERVABLE MARKOV DECISION PROCESS

A partially observable Markov decision process (POMDP) is a variation of the MDP that provides a framework for planning under uncertainty in stochastic environments. Instead of the agent knowing the true state of the system, the agent receives observations, which may or may not be synonymous with the true state of the system, and reasons in a state of beliefs when making decisions [18]. To account for uncertainty and the possible disparity between observation and state, observations (Z) as well as observation probabilities (O) are added to the POMDP [18]. The collection of components of a POMDP vary slightly from an MDP and are represented as $\{S, A, Z, T, O, R\}$ [18].

The components S, A, T and R are identical to those in an MDP, while the components, Z and O, are specific to POMDPs [18]. When an action is taken, the system transitions from one state to another according to its transition probabilities and receives an observation $(z \in Z)$ with probability $O(s', a, z) = p(z|s', a)$ and a reward $R(s, a)$. Because the state cannot be directly observed in a POMDP, a belief space (B) is maintained and represented as a probability distribution over S [19].

The policies of POMDPs also vary from those in MDPs. The policy (π) in a POMDP is a mapping from the belief space to actions instead of a mapping from states to actions found in MDPs and is represented by $\pi = B \rightarrow A$ [12]. It determines the action at belief b_t by $\pi(b) \in A$ [13].The belief space is a probability distribution over S that includes a history of action and observation pairs and is updated after every decision epoch $b_t = \tau(b_{t-1}, a_t, z_t)$ [13]. The values of policies are also different in POMDPs and are represented by

$$V^{\pi}(b) = E\left(\sum_{t=0}^{\infty} \gamma^t R\big(s_t, \pi(b_t)\big) \mid b_0 = b\right) \text{[13].} \tag{5}$$

The goal of POMDPs, just as with MDPs, is to maximize long term total discounted reward, and one method used to solve them is known as value iteration. Value iteration is reasoning back in time to determine the sequence of optimal actions instead of working optimally towards a goal [13]. Value iteration is performed using a modified version of the Bellman equation that was used to solve MDPs:

$$V^*(b) \leftarrow \max_{a \in A}\left\{\sum_{s \in A} b(s)R(s,a) + \gamma \sum_{z \in Z} p(z|b,a)V^*\big(\tau(b,a,z)\big)\right\} \text{[13].} \tag{6}$$

With large POMDPs, this solution becomes computationally intractable due to the "curse of dimensionality" and the "curse of history" [13]. The "curse of dimensionality" refers to the size of the belief space growing exponentially as the number of state variables grows, and the "curse of history" refers to the action-observation histories growing exponentially with the planning horizon [13].

E. POMDP SOLUTION METHODS

Offline planning and online planning are the two primary approaches to solving POMDPs [13]. There are difficulties in scaling to large POMDPs with offline planning because they must plan for all beliefs and future contingencies when determining the optimal policy, which yields extremely large search spaces that are difficult to work with [13]. With online planning, the search space is significantly reduced because planning takes place locally due to the optimal action being chosen for only the current belief through a

lookahead search in the neighborhood of that local belief [13]. Because online planning works with smaller state spaces, it is able to solve much larger POMDPs. This thesis relies on the concepts of online planning for solving POMDPs.

In online planning, the agent starts in an initial belief state, b_0, and searches for an optimal action, a^* [13]. Once the agent takes the action, it receives a new observation, and updates its belief state based on its most recent action observation pair [13]. At every decision epoch, the belief space is updated using Bayes' rule by recording action observation pairs according to

$$b_t(s') = \eta \cdot O(s', a_t, z_t) \sum_{s \in S} T(s, a_t, s') \cdot b_{t-1}(s) \text{ [13]}, \tag{7}$$

where η is a normalizing constant and the belief at time t is represented by b_t.

There are multiple methods of conducting online planning to solve POMDPs including Monte Carlo sampling, heuristic search, and branch-and-bound pruning [13]. The disadvantage to most online planning algorithms is that they represent the belief as a probability over the state space, which limits the size of POMDPs they can solve [13]. Both Partially Observable Monte-Carlo Planning (POMCP) and DESPOT are online planning algorithms that rely heavily on Monte Carlo sampling; however, both algorithms avoid this disadvantage by representing their beliefs as a set of sampled states [13]. POMCP is unique in that it uses a Monte Carlo tree search (MCTS), which is a heuristic search algorithm, on a belief tree to find the optimal policy [13]. This is good for large POMDPs because the heuristic nature of the search algorithm significantly reduces the search space, but the results can be inaccurate due to overfitting caused by being misguided by the upper confidence bound of the heuristic [13]. DESPOT is very similar to the POMCP approach; however, it incorporates all three methods of online planning and alleviates overfitting by the use of regularization [13]. DESPOT is the method used to solve POMDPs in this thesis.

F. DESPOT

DESPOT is a sparse approximation of standard belief tree that focuses online planning on capturing the execution of all policies using a set of randomly sampled scenarios, *K*, rather than all possible scenarios [13]. A scenario is a unique trajectory of the

execution of a policy that is determinized by a random set of numbers [13]. A DESPOT tree contains only the action observation histories under K sampled scenarios but converges to the standard belief tree as $K \to \infty$ [13]. By choosing K scenarios instead of all scenarios, DESPOT samples both states and observations from a belief tree, preventing each from growing exponentially, which alleviates both the "curse of dimensionality" and the "curse of history" [13]. The policy obtained by using DESPOT is proven to be near-optimal and approaches the optimal policy as $K \to \infty$, where K is defined as

$$K \in O\big(|\pi|\ln(|\pi||A||Z|)\big) \text{ [13].} \tag{8}$$

A picture of both a standard belief tree and a DESPOT are shown in Figure 5. The starting node on a DESPOT belief tree represents the initial belief and every following node represents updated belief states [13]. Each belief node branches into action edges that further branch into observation edges leading to a new belief state. DESPOT performs a lookahead search on the tree at the initial belief for a policy that maximizes the value function in order to find the optimal action [13]. At each belief node on the tree, the Bellman equation is applied to maximize the value function to find the optimal policy [13]. A post order traversal is performed on the belief tree, solving the Bellman equation at each node [13]. This method is very similar to value iteration as it is working backwards from the bottom of the tree to find the maximum value at each node that will lead to an optimal action for b_0.

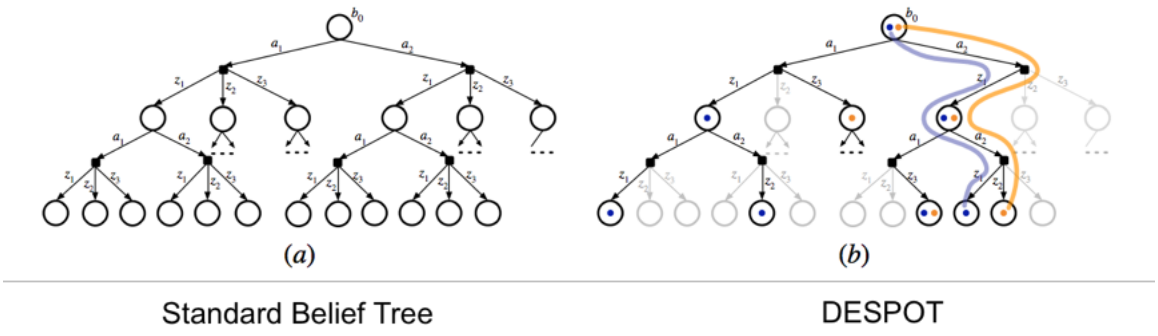

Figure 5. Standard Belief Tree and DESPOT. Source: [13].

Two notable features of DESPOT are that it uses sampling and anytime heuristic search [13]. Anytime heuristic search, which constructs the DESPOT incrementally under a heuristic, speeds up lookahead search and allows the algorithm to output the highest valued action with a partially constructed DESPOT under limited planning time [13]. Because DESPOT speeds up lookahead search, which sometimes overfits sampled scenarios resulting in a non-optimal policy, DESPOT employs regularization to balance the estimated value of a policy using sampled scenarios and policy size [13]. These characteristics together give DESPOT an advantage over other online planning algorithms and are the reason it was chosen as the solution method in this thesis.

G. SUMMARY

In summary, we discussed the importance of the intrusion kill chain, the advantages of moving target detection techniques, Markov decision processes to include POMDPs, and widely used MDP and POMDP solution methods to include the anytime online algorithm, DESPOT. Each of these topics are integral to understanding the research presented in this thesis.

Chapter III will discuss the method used to model our attack and defense scenarios involving the intrusion kill chain and moving target defense techniques as well as the method used to optimize the implementation of MTD techniques.

THIS PAGE INTENTIONALLY LEFT BLANK

III. OPTIMIZATION OF MOVING TARGET DEFENSE TECHNIQUES

A background of the intrusion kill chain, MTD, Markov decision processes, partially observable Markov decision processes, and DESPOT were discussed in Chapter II. Here we will discuss the cyber defense scenario on which our model is based as well as the method used to develop that model and solve it with the objective of optimizing the moving target defense techniques within it.

A. OPTIMIZATION METHOD

In order to develop a method of optimizing MTD techniques, it is necessary to define the cyber defense scenario on which our model is based. We aim to model the defense of a protected network against the progression of a cyber-attack and to analyze that defense in terms of cost/benefit of the defender, similar to [6]. The dynamic defense model describing our scenario is pictured in Figure 6 and is very closely related to the model described in [20]. In the case that a complex cyber-attack occurs, such as an APT attack, the attacker will work within the cyber network to attack a protected network, which typically includes an intrusion detection system (IDS) [20]. Once the protected network is breached, the IDS system is alerted and sends indications to the protected system, or defender, to initiate defenses, or actions [20].

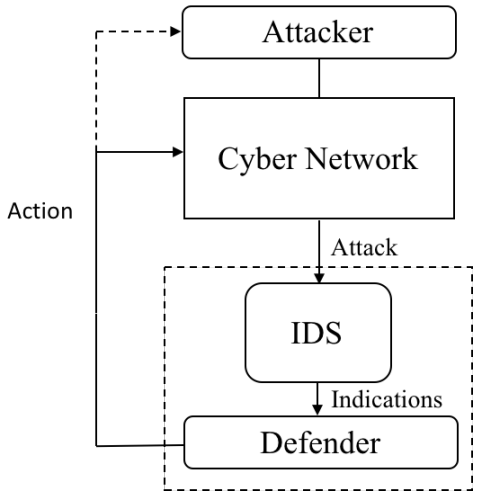

Figure 6. Dynamic Defense Model. Adapted from [20].

In this thesis, the attacker works through the cyber network or environment to attack a defender, after which the defender initiates actions in defense according to observations it receives. We do not include an IDS of the typical nature in this thesis; however, the model we use provides a way for the defender to receive observations concerning the state of its network similarly to how an intrusion detection system provides indications. The attack resembles a cyber-attack that follows the framework of the intrusion kill chain, and defensive actions resemble MTD techniques. The MTD techniques interrupt the cyber-attack in order to impede the progress of the attack and defend the protected network. In order to model this cyber defense scenario in such a way that allows for the defender actions to be optimized in an uncertain environment, we will collect data and explore possible modeling solutions.

The method by which the cyber defense scenario is modeled and solved with the objective of optimizing MTD techniques is depicted in Figure 7. This method is comprised of four steps: model development, model solution, analysis of results, and an evaluation of whether or not optimization is achieved. If optimization is achieved, the MTD techniques have been optimized and the method ends; however, if optimization is not achieved, it is recommended to return to model development and repeat the method with modifications to each step until optimization is achieved.

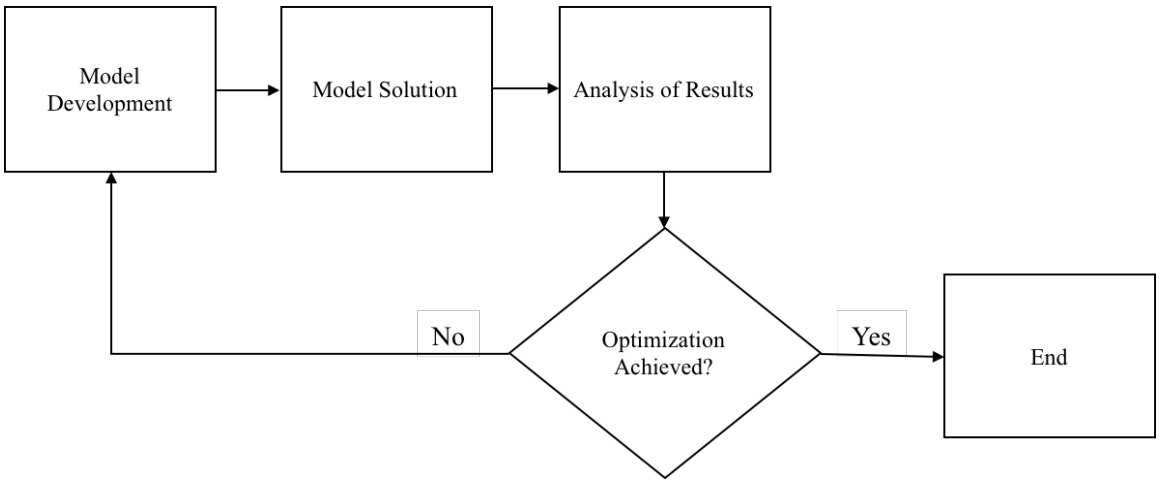

Figure 7. Method of Approach

Each of the steps in the method of approach are integral in optimizing MTD techniques. In model development, we aim to develop a model that incorporates every aspect of the cyber defense scenario. In model solution, we choose an existing solution method that is best fit to solve our model. In analysis of results, we plan to analyze the results from the solution method to determine how well our developed model and solution method lead to meeting our objectives. With those results, we will be able to determine if the MTD techniques are optimized in our model. If the techniques are not optimized, optimization improvement can be performed by going back to model development and modifying each of the subsequent steps. In this thesis, optimization improvement involves ideas for future work to further improve the results.

B. MODEL DEVELOPMENT

The model development for this thesis is three-fold: creating an attack model that depicts the steps involved in network intrusion followed by an attack, creating a defense model that includes network level MTD techniques, and creating a POMDP model that combines both the attack and defense models. Both the attack and defense models developed adhere very closely to the model employed in [6]. Our attack model depicts the process an attacker typically follows before launching a successful attack on a defender and includes five distinct states: start, target scan, vulnerability scan, exploit launch, and attack [4]. Our defense model includes two types of MTD techniques tailored to network

defense. In theory, moving target defense techniques can be implemented at any point in our attack model in order to delay the attacker by rendering some of his/her collected information obsolete [4]. In this thesis, the state of the defender system will correlate with the state the attacker is in because the state of the defender system from the defense perspective is the extent to which he/she is being attacked.

1. Attack Model

The attack model we propose in this thesis adheres very closely to the intrusion kill chain as well as to the model developed in [4]. The sequence of states for our attack model is depicted in Figure 8. The start state is a neutral state in which the attacker is not actively trying to gain any information on or attack the defender. If the attacker starts the process of launching an attack, it will transition to the target scan state. The target scan state is where the attacker conducts a survey of the defender ports in search of ones that are open to accepting packets and facilitating some type of network service [4]. If the attacker finds one or more open ports, it transitions to the vulnerability state, which is where the attacker surveys the services running on those ports in search of ones that are vulnerable [4]. If the attacker is successful in finding a vulnerable service, it transitions to the exploit launch state, which is where the attacker attempts to gain a connection with that service [4]. Finally, if the attacker is successful in connecting to a vulnerable service, and therefore the defender, it transitions to the attack state [4]. The attack state is where the attacker utilizes its connection to the host through the server in order to launch an attack and compromise the defender system [4].

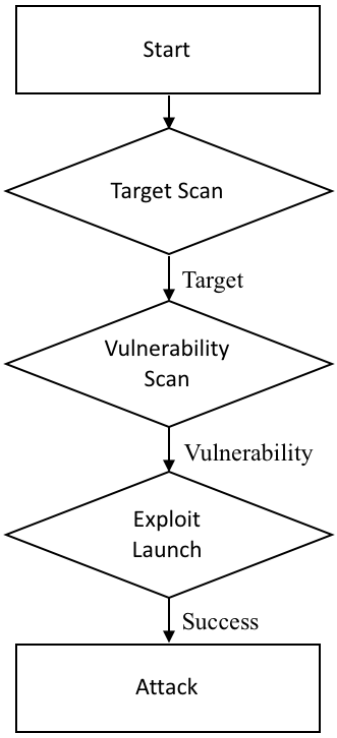

Figure 8. Attack Sequence. Adapted from [4].

Similar to [6], we built a Markov model of attack progress, aligning states to the phases of attack in [4]. Our attack model, depicted in Figure 9, is a Markov chain including the states depicted in Figure 8. The states, S_0, S_1, S_2, S_3, S_4, stand for start, target scan, vulnerability scan, exploit launch, and attack, respectively. If the attacker is successful in each of the states, the attack progression will proceed directly from state S_0 to S_4; however, if the attacker fails to achieve its goal in any of the states, it will remain in the same state before continuing towards S_4 as shown in Figure 8. Figure 9 depicts the state transitions as transition probabilities, where p_{01}, for example, represents the probability of transitioning from state 0 to 1.

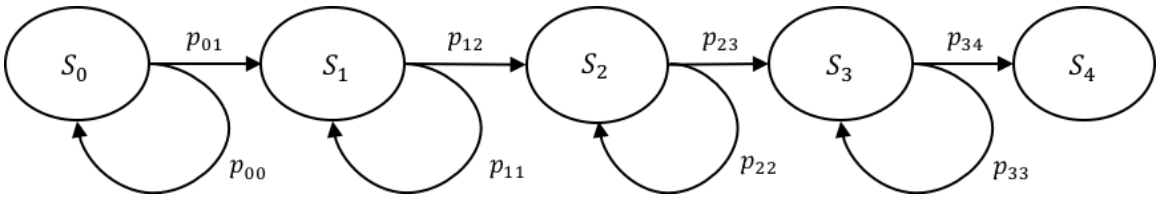

Figure 9. Attack Model. Adapted from [6]

2. Defense Model

The attack model in Figure 9 provides a framework on which MTD techniques can be implemented in order to delay or completely thwart an attack. The two MTD techniques we use in our defense model are IP reconfigurations and service reconfigurations, both of which affect the network dynamics of a protected system. IP reconfigurations randomize the IP address of a system that make port scanning impossible for the attacker until it regains the new IP address [4]. Service reconfigurations randomize the services a system is using that can render the vulnerable services that the attacker is using to compromise it obsolete [4]. Both reconfigurations make it difficult for the attacker to succeed; however, IP reconfigurations are tailored towards the target scan state and service reconfigurations are tailored towards the vulnerability state [4]. A Markov chain of the complete attack and defense model is depicted in Figure 10. If an IP reconfiguration occurs and is successful, and the model is in the S_1 state, the model will transition back to S_0 with the transition probability p_{10}. Likewise, if a service reconfiguration occurs and is successful, and the model is in the S_2 state, the model will fall back to S_1 with the transition probability p_{21}.

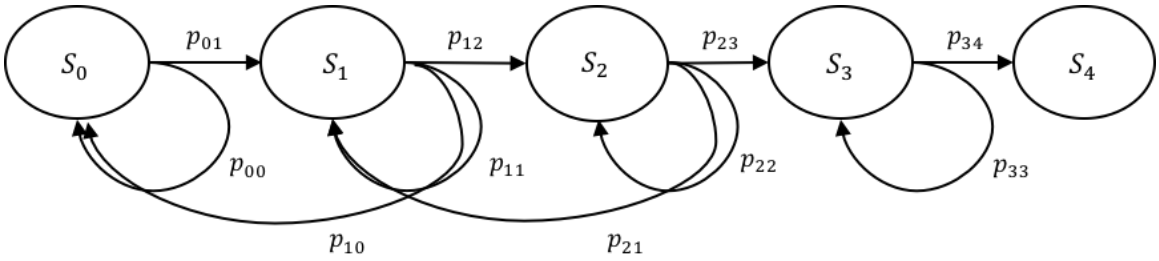

Figure 10. Attack/Defense Model. Adapted from [6].

3. POMDP

We propose a POMDP model in this thesis in order to facilitate the attack/defense model in Figure 10 as well as to facilitate optimal decision making in an uncertain environment. A schematic depicting the components of a POMDP and how the components relate to one another is shown in Figure 11. In a POMDP process, the agent makes decisions based on policies (π) and the belief state (B), and these decisions result in actions that affect the both the environment in which the agent is residing, defined by states (S) and transition probabilities, and the agent in the form of a reward/cost due to the effect the action has on its own system [11]. Every time an action is taken, the belief state is updated based on the previous action-observation pair and the process repeats [11].

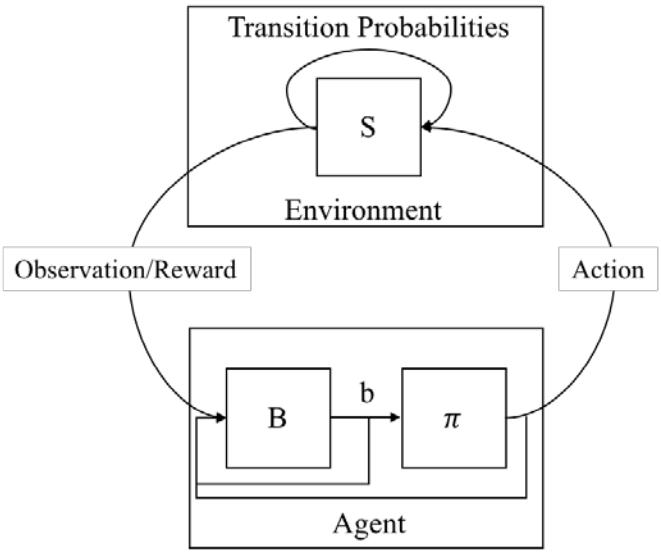

Figure 11. Schematic Diagram of a General POMDP Model. Adapted from [11].

The schematic of the proposed POMDP model for this thesis is depicted in Figure 12, and it follows the organization of the schematic shown in Figure 11. In our POMDP model, the agent is referred to as a defender who takes defensive actions to protect its network. The environment resembles a cyber network in which the attacker is able to reach the defender. The defender actions are carried out in the form of MTD techniques and are based on the observations the defender is receiving as well as the history of action-

observation pairs contained in the belief state, B. These actions affect the progression of the attack and therefore the cyber network, which in turn affects the defender in terms of both observations and costs. The costs to the defender are in the form of seconds detracting from system availability due to the time it takes to conduct MTD techniques.

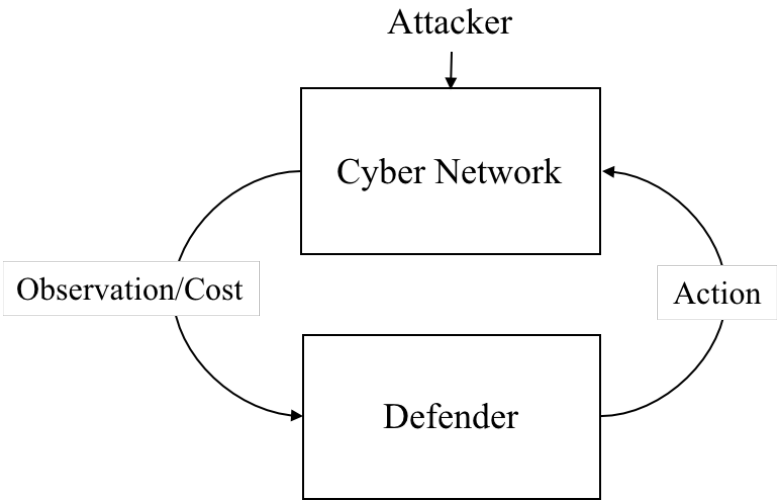

Figure 12. Schematic Diagram of the Proposed POMDP Model. Adapted from [6].

The POMDP model we present in this thesis is an infinite-horizon discounted POMDP. It is infinite-horizon in that it continues in time without a defined ending point, and it is discounted in that rewards in the future are valued less than present awards according to the discount variable, γ [9]. The Markov chain for our infinite-horizon discounted POMDP model is depicted in Figure 13. In addition to the attack/defense model depicted in Figure 10, it includes attack recovery delay states to resemble attack recovery time and to allow the model to continue to cycle without a defined ending point, as well as additional transition probabilities to reflect that the attack is not limited to progressing sequentially [6]. The transition matrix of the model depicted in Figure 13 can be written as

$$P_T = \begin{bmatrix} p_{00} & p_{01} & p_{02} & p_{03} & 0 & 0 \\ p_{10} & p_{11} & p_{12} & p_{13} & 0 & 0 \\ 0 & p_{21} & p_{22} & p_{23} & 0 & 0 \\ 0 & 0 & 0 & p_{33} & p_{34} & 0 \\ 0 & 0 & 0 & 0 & 0 & p_{4R} \\ p_{R0} & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \quad (9)$$

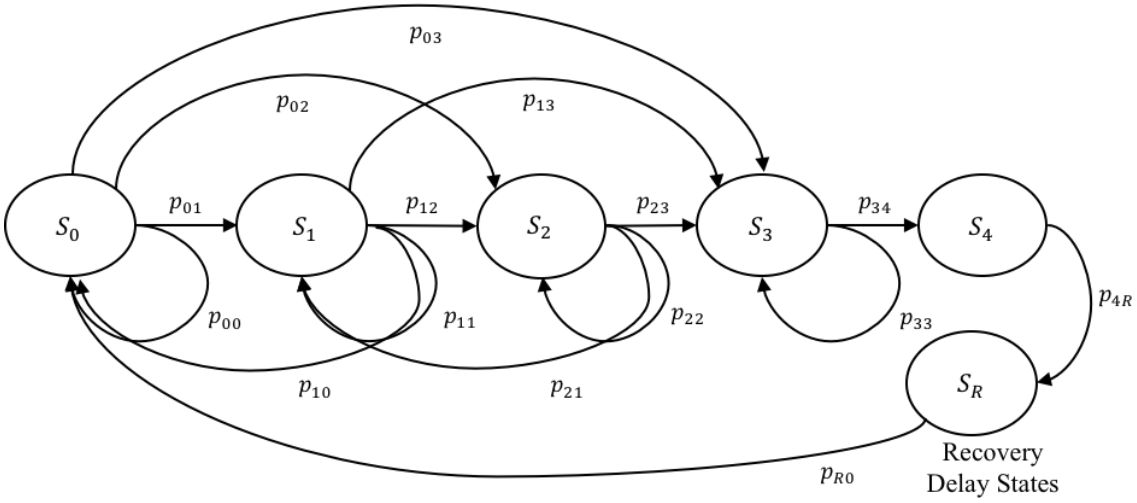

Figure 13. The State-Transition Diagram of the Proposed POMDP Model. Adapted from [6].

The state variables and observation variables in this POMDP model are very similar to one another. The states include S_0 through S_4, as well as the recovery delay states. The observation variables only include S_0 through S_4. In our POMDP model, the state of the system is observed by the defender, and the attack recovery delay states are all observed as an attack.

The action variables and cost variables in our POMDP model are closely related. The actions represent the MTD techniques described as part of the defense model pictured in Figure 10 and include three actions: nil, IP reconfiguration, and service reconfiguration. The action nil is when the defender conducts neither an IP reconfiguration or a service reconfiguration. Cost variables are directly related to the action variables as each of the actions, except for nil, take time to conduct and come with a cost to the defender in terms of network downtime. A cost is also incurred by the defender when an attack occurs, and

it represents the idea that network downtime could result from an attack in a cyber-defense scenario.

The discount factor, γ, describes how immediate costs are valued in comparison with future costs when calculating the long term total discounted reward as shown in Equation 4 [9]. We propose using a large discount variable close to 1 because, although preventing an attack in the present is most important, we consider preventing both present and future attacks in our cyber-defense scenario and model to be vital for the optimization of moving target defense techniques.

The transition probabilities, p_{ij}, are action dependent conditional probabilities that describe the state transitions of the POMDP model [11]. The transition probabilities of our POMDP model will be calculated using the data and results presented in [5]. The work presented in [5] describes the likelihood that a port scan or a vulnerability scan precedes an attack. These likelihoods directly relate to our attack model as our attack model includes target scans and vulnerability scans. In this thesis, a target scan is synonymous with a port scan. Once calculated, the transition probabilities used in our model stay the same throughout the course of our research.

The observation probabilities in our POMDP model describe the likelihood that the defender is able to observe the true state of the system, and these probabilities change throughout cyber activity and form the basis for different variations of our POMDP model. Similarly to transition probabilities, the observation probabilities are contained within an observation matrix where the columns represent the observations and the rows represent the states. By changing the observation probabilities, it is possible to determine how well the moving target defense techniques are optimized in both fully observable cases and partially observable cases with varying levels of uncertainty.

C. MODEL SOLUTION

We propose DESPOT as the solution method for the POMDP model because of its notable features that make it a superior online planning algorithm as described in Chapter II [6]. In order to achieve our objectives, we plan to first solve the POMDP model with full observability to obtain the optimal policy and then to solve modifications of that POMDP

model with uncertainty using DESPOT as implemented in [21]. We create three different modifications of the POMDP model with full observability by changing the observation matrix each time to represent three different scenarios in which detection or miss detection is more likely.

In order to obtain the best possible results with DESPOT, we plan to test a few of the many default parameters belonging to the algorithm before we solve any of the POMDP models. Many of the parameters may be altered by the user in order to best adapt the algorithm to the POMDP being solved, and the parameters that are of interest to us are summarized in Table 1. The parameters in Table 1 include Decision Time, Search Depth, and Particles, each with their respective default values and descriptions [21]. Decision time is the time allotted for each decision epoch, and it is crucial that there is enough time for the algorithm to make the best decision. Search depth refers to the length of the search tree and describes how far into the future the algorithm will lookahead. Particles describe the size of the belief state and how many observation and action pairs from the history of all pairs will be considered for future decisions. Each of these parameters are very important to this research as they are key elements in solving the POMDP model as well as key elements that we might need to change for DESPOT to work favorably in our research.

Table 1. DESPOT Parameters of Interest. Adapted from [21].

Parameter	Default Value	Description
Decision Time (seconds)	1	Time allotted for each decision
Search Depth	90	Depth of the search tree
Particles	500	Particles used to represent the belief state

D. ANALYSIS OF RESULTS

Once we solve each of our POMDP models with DESPOT, we will use the concepts of optimal policy, limiting state probabilities, attack percentage, cost, and availability to analyze our results. The optimal policy of the POMDP model with full observability will serve as a control model and allow for the comparison of each of the modified POMDP models. In order to understand the theoretical percentage of attacks that could be prevented if defenses are used optimally, we will use the concept of limiting state probabilities. This will allow us to have a baseline for determining how well the defenses are employed in each of the POMDP models tested with DESPOT. Finally, the parameters of percentage of attacks prevented, cost in terms of network downtime, and network availability will all be used to determine if the advantages of MTD techniques were being maximized and undue cost was minimized.

E. SUMMARY

In this chapter, we discussed the cyber defense scenario on which our research is based as well as the method we use to optimize MTD techniques to include model development, model solution, and analysis of results. Chapter IV will discuss the implementation of this method and the analysis of results.

IV. RESULTS

The methodology described in Chapter III is implemented in this chapter. This chapter is divided into two sections. The first section describes the POMDP model, its optimal policy, and limiting state probabilities, and the second section details the simulation and presents the results.

A. POMDP MODEL

The POMDP model developed in this thesis follows the format of an infinite-horizon discounted POMDP as described in Chapter III. Appendix A contains the POMDP xml file that formats the model for ingest into DESPOT, developed from the template provided by [21]. This section details each of our POMDP model components, including $S, A, Z, T, O, R,$ and γ, as well as a summary of those components and the optimal policy for the POMDP.

1. Variables

The set of states and observations in our POMDP model are identical to those described as part of the POMDP model shown in Figure 13, with the addition of expanding the set of states to include distinct recovery delay states for added time delay within the model. We expanded the attack recovery delay states into 4 separate states, S_5 through S_8, as depicted in Figure 14. In this thesis, the attack recovery delay states incur no cost to the system and all have transition probabilities of 1. It is the transitions themselves between the attack recovery delay states that cause the time delay in our model, representing the idea that it takes a network time to recover from an attack. Each transition, regardless of the transition probabilities belonging to them, requires time to execute. By adding four attack recovery delay states, we were able to create this time delay while ensuring that it did not outlast the attack progression itself.

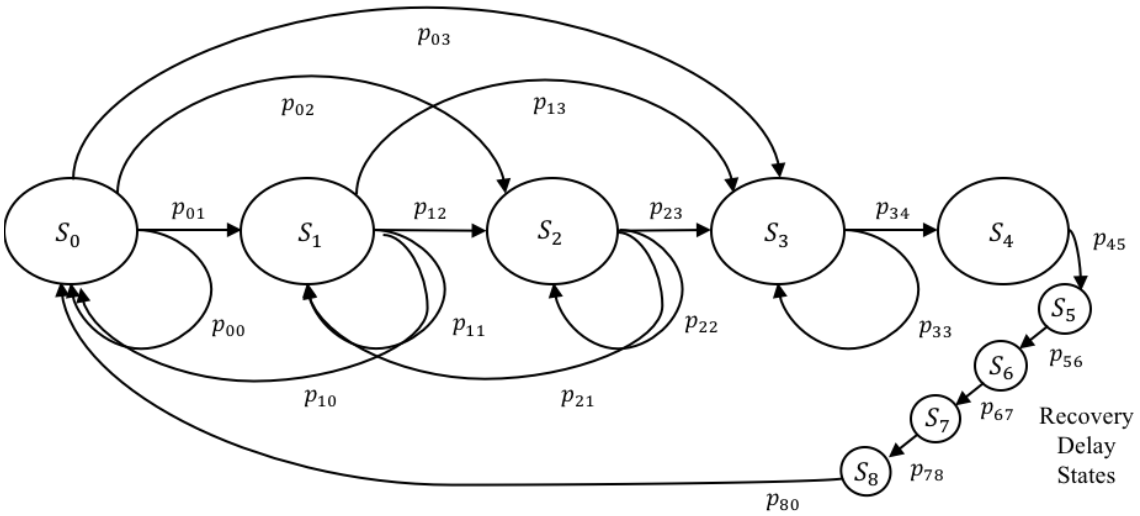

Figure 14. POMDP Model with Recovery Delay States.
Adapted from [6].

The set of actions in our POMDP model are identical to those described as part of the POMDP model in Figure 13. The rewards for our POMDP model exist in the form of cost in seconds to the system that affect the availability of the network. The three actions in our POMDP model include nil, IP reconfiguration, and service reconfiguration, all of which are associated with costs that were developed using the research presented in Connell et al. [4]. The costs of the actions, nil, IP reconfiguration, and service reconfiguration, are 0 seconds, -0.635 seconds, and -9.95 seconds, respectively [4]. There is also a cost incurred to the system when it is attacked to represent the impact an attack would have on a protected cyber network in terms of recovery time. We chose a cost of -695 seconds for each attack because the cost of an attack needed to be much larger than either an IP or service reconfiguration to emphasize its impact on the system. However, we also ensured the cost of an attack was not so large that IP and service reconfigurations would be employed constantly due to their costs being insignificant compared to an attack. Each of the costs employed in this thesis were chosen in order to allow the system to reach an optimized performance.

We chose γ to be 0.95 for our POMDP model. With a high discount variable of 0.95, both immediate and future rewards were highly valued, with immediate rewards being valued only slightly higher than future rewards.

2. Transition Probabilities

In this thesis, the transition probabilities describe both the likelihood that the attacker will or will not proceed forward through the stages of the intrusion kill chain and the likelihood that the attacker will have to take a step back if defensive action is taken. The transition matrix for the POMDP model in Figure 14 is written as:

$$P_T = \begin{bmatrix} p_{00} & p_{01} & p_{02} & p_{03} & p_{04} & p_{05} & p_{06} & p_{07} & p_{08} \\ p_{10} & p_{11} & p_{12} & p_{13} & p_{14} & p_{15} & p_{16} & p_{17} & p_{18} \\ p_{20} & p_{21} & p_{22} & p_{23} & p_{24} & p_{25} & p_{26} & p_{27} & p_{28} \\ p_{30} & p_{31} & p_{32} & p_{33} & p_{34} & p_{35} & p_{36} & p_{37} & p_{38} \\ p_{40} & p_{41} & p_{42} & p_{43} & p_{44} & p_{45} & p_{46} & p_{47} & p_{48} \\ p_{50} & p_{51} & p_{52} & p_{53} & p_{54} & p_{55} & p_{56} & p_{57} & p_{58} \\ p_{60} & p_{61} & p_{62} & p_{63} & p_{64} & p_{65} & p_{66} & p_{67} & p_{68} \\ p_{70} & p_{71} & p_{72} & p_{73} & p_{74} & p_{75} & p_{76} & p_{77} & p_{78} \\ p_{80} & p_{81} & p_{82} & p_{83} & p_{84} & p_{85} & p_{86} & p_{87} & p_{88} \end{bmatrix} \quad (10)$$

We calculated the transition probabilities for the POMDP model in Figure 14 based on the data presented in both [4] and [5], similarly to the method used in [6]. The data presented in [5] was collected in order to determine if port scans, ICMP scans, and vulnerability scans are precursors to an attack [5]. In this thesis, we group port scans and ICMP scans into one group called target scans because an IP reconfiguration can derail the progress of either of these types of scans. The data we used from [5] includes attack data from a 48-day study where both management traffic, which is accepted as normal traffic, and malicious activity were observed on a subnet including 2 target computers [5]. First, each type of scan was classified by the number of packets involved in the connection in order to be able to distinguish between the different scans [5]. There was a total of 908,963 packets of malicious and management traffic analyzed, and of those packets, 59,468 were determined to be malicious [5]. Of the malicious packets, 22,170 were found to include ICMP scans, port scans, vulnerability scans, and attacks directed towards the target computers, 20,675 of which were precursors to an attack [5]. It was also determined that there was a total of 760 attacks that occurred throughout this study on the two target computers, 50% of which were preceded by port, ICMP, or vulnerability scans. This data is summarized in Table 3.

Table 2. Summary of Total Malicious Packets and Scans. Adapted from [5].

Total packets observed	908,963
Total malicious packets	59,468
Total port, ICMP, and vulnerability scans	22,170
Total scans leading to an attack	20,675
Total attacks	760
Total attacks preceded by scans	380

The findings presented in [5] are grouped into two sections: one describing the number of different scans preceding attacks and the other describing the number of attacks preceded by different types of scans [5]. We summarized this data into Venn diagrams that are pictured in Figures 15 and 16. The Venn diagram in Figure 15 depicts the number of distinct port, ICMP, and vulnerability scans as well as the combination of these scans that preceded attacks. The Venn diagram in Figure 16 depicts the number of attacks that were preceded by distinct scans as well as combinations of scans. For the purpose of our research, we combined the port and ICMP scans into one group of target scans in the Venn diagrams on the right in both figures, as IP reconfigurations target both port and ICMP scans.

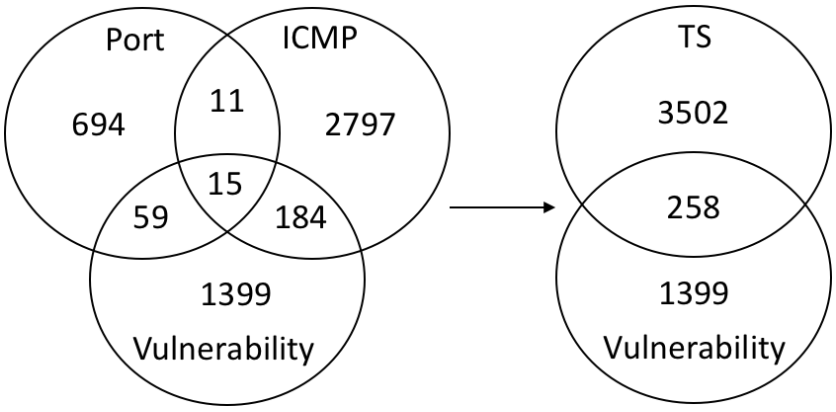

Figure 15. Scans Preceding Attacks. Adapted from [5].

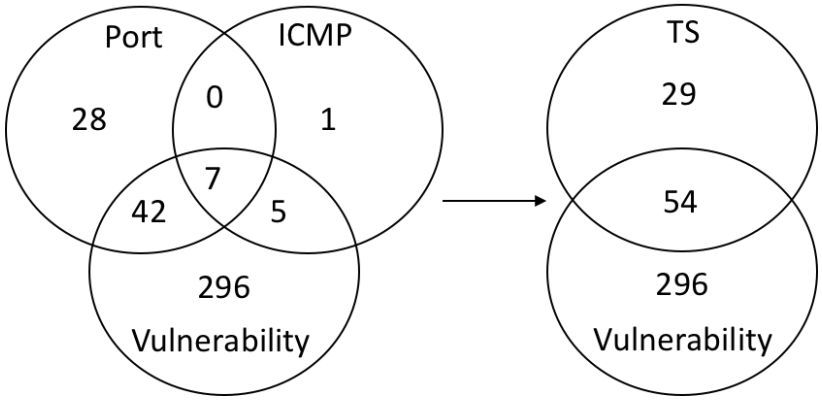

Figure 16. Number of Attacks Preceded by Scans. Adapted from [5].

We calculated the transition probabilities that account for the effect of defensive action being taken, such as IP reconfigurations and service reconfigurations, using the data presented in [4]. The experiments conducted in [4] to determine how successful MTDs were in thwarting attacks were run under the constraints that there were 256 available IP addresses, similar to a /24 network, and that there were three web services available to the host [4]. In this thesis, we operated under the same constraints assuming that there were 256 available IP addresses and 3 web services available to the defender. Thus, for our POMDP model, we calculated $p_{10,IR}$ to be 255/256 and $p_{21,SR}$ to be 2/3 [4]. These probabilities account for the possibility of the attacker choosing the exact same IP address or service that the defender "randomly" switched to during a reconfiguration. Although IP and service reconfigurations are transparent to the defender conducting them, these reconfigurations appear random to the attacker.

As transition probabilities vary according to actions, in our POMDP model, $p_{ij,N}$ represents probabilities when no action is taken, $p_{ij,IR}$ represents probabilities when an IP reconfiguration occurs, and $p_{ij,SR}$ represents probabilities when a service reconfiguration occurs. There are four transition probabilities stemming from the start state: $p_{00}, p_{01}, p_{02}, p_{03}$. Neither service nor IP reconfigurations affect the start state of the attack model; thus, the probabilities stemming from the start state remain the same even if a reconfiguration is conducted while in the state. The start-state transition probabilities were calculated as follows [5]:

$$p_{00,N} = p_{00,IR} = p_{00,SR} = 1 - \varepsilon_{s:mp}\,, \tag{11}$$

$$p_{01,N} = p_{01,IR} = p_{01,SR} = \frac{\left(\varepsilon_{ts} \times \varepsilon_{s:mp}\right)}{\left(\varepsilon_{ts} + \varepsilon_{ts^c} + \varepsilon_{a|s}\right)}\,, \tag{12}$$

$$p_{02,N} = p_{02,IR} = p_{02,SR} = \frac{\left(\varepsilon_{ts^c} \times \varepsilon_{s:mp}\right)}{\left(\varepsilon_{ts} + \varepsilon_{ts^c} + \varepsilon_{a|s}\right)}\,, \tag{13}$$

$$p_{03,N} = p_{03,IR} = p_{03,SR} = \frac{\left(\varepsilon_{a|s} \times \varepsilon_{s:mp}\right)}{\left(\varepsilon_{ts} + \varepsilon_{ts^c} + \varepsilon_{a|s}\right)}\,, \tag{14}$$

where $\varepsilon_{s:mp}$ is the ratio of total scans preceding an attack (20675) to total malicious packets (59,468), ε_{ts} is the total target scans preceding attacks (3760), ε_{ts^c} is the unique vulnerability scans preceding attacks (1399), and $\varepsilon_{a|s}$ is the total attacks preceded by scans (380).

There are four transition probabilities stemming from the target scan state: p_{10}, p_{11}, p_{12}, p_{13}. Because IP reconfigurations affect the target scan state of the attack model, the transition probabilities stemming from the state are also affected. To calculate the transition probabilities that account for IP reconfigurations, we multiplied the nil transition probabilities by p_{10} [4]. The target scan state transition probabilities were calculated as follows [5]:

$$p_{10,N} = p_{10,SR} = 0\,, \tag{15}$$

$$p_{12,N} = p_{12,SR} = \frac{\varepsilon_{ts \cap vs}}{\varepsilon_{ts}}\,, \tag{16}$$

$$p_{13,N} = p_{13,SR} = \frac{\varepsilon_{a|vs^c}}{\varepsilon_{ts}}\,, \tag{17}$$

$$p_{11,N} = p_{11,SR} = 1 - \left(p_{12,N} + p_{13,N}\right), \tag{18}$$

$$p_{10,IR} = \frac{255}{256}, \tag{19}$$

$$p_{11,IR} = \left(1 - p_{10,IR}\right) \times p_{11,N}\,, \tag{20}$$

$$p_{12,IR} = \left(1 - p_{10,IR}\right) \times p_{12,N}\,, \tag{21}$$

$$p_{13,IR} = \left(1 - p_{10,IR}\right) \times p_{13,N}\,, \tag{22}$$

where $\varepsilon_{ts \cap vs}$ is the intersection of target scans and vulnerability scans preceding attacks (258), ε_{ts} is the total target scans preceding attacks (3760), and $\varepsilon_{a|vs^c}$ is the total attacks preceded by unique target scans (29).

There are three transition probabilities stemming from the vulnerability scan state: p_{21}, p_{22}, p_{23}. Because service reconfigurations affect the vulnerability scan state of the POMDP model, the transition probabilities stemming from the state are also affected. To calculate the transition probabilities that account for service reconfigurations, we multiplied the nil transition probabilities by p_{21} [4]. The vulnerability state transition probabilities were calculated as follows [5]:

$$p_{21,N} = \; p_{21,IR} = 0\,, \tag{23}$$

$$p_{22,N} = \; p_{22,IR} = \frac{(\varepsilon_{vs} - \varepsilon_{a|vs})}{\varepsilon_{vs}}\,, \tag{24}$$

$$p_{23,N} = \; p_{23,IR} = \frac{\varepsilon_{a|vs}}{\varepsilon_{vs}}\,, \tag{25}$$

$$p_{21,SR} = \frac{2}{3}\,, \tag{26}$$

$$p_{22,SR} = \left(1 - p_{21,SR}\right) \times p_{22,N}\,, \tag{27}$$

$$p_{23,SR} = \left(1 - p_{21,SR}\right) \times p_{23,N}\,, \tag{28}$$

where $\varepsilon_{a|vs}$ is the total attacks preceded by vulnerability scans (350), and ε_{vs} is the total vulnerability scans preceding attacks (1657).

There are two transition probabilities stemming from the attack state: p_{33}, p_{34}. Because neither service nor IP reconfigurations affect the attack state of the attack model, the probabilities stemming from this state remain the same even if a reconfiguration is conducted while the model is in the attack state. The exploit launch state transition probabilities were calculated as follows [5]:

$$p_{33,N} = p_{33,IR} = p_{33,SR} = \frac{\varepsilon_{a|s}}{\varepsilon_a}, \tag{29}$$

$$p_{34,N} = p_{34,IR} = p_{34,SR} = 1 - p_{33,N}, \tag{30}$$

where $\varepsilon_{a|s}$ is the total attacks preceded by scans (380), and ε_a is the total attacks (760).

All of the calculated transition probabilities used in our POMDP model are summarized in three 9×9 matrices grouped according to action: $P_{T,N}$, $P_{T,IP}$, $P_{T,S}$. The transition matrices are given by

$$P_{T,N} = \begin{bmatrix} 0.6523 & 0.2360 & 0.08780 & 0.02391 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.9237 & 0.06862 & 0.007713 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.7888 & 0.2112 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{31}$$

$$P_{T,SR} = \begin{bmatrix} 0.6523 & 0.2360 & 0.08780 & 0.02391 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.9237 & 0.06862 & 0.007713 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.6667 & 0.2629 & 0.07041 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{32}$$

$$P_{T,IR} = \begin{bmatrix} 0.6523 & 0.2360 & 0.08780 & 0.02391 & 0 & 0 & 0 & 0 & 0 \\ 0.9956 & 0.004087 & 0.0003036 & 0.00003413 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.7888 & 0.2112 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \quad (33)$$

A diagram of the Markov chain for our POMDP model annotated with all of the calculated transition probabilities is depicted in Figure 17. The probabilities are color coded according to each type of reconfiguration: black represents the nil action, blue represents the service reconfiguration action, and red represents the IP reconfiguration action.

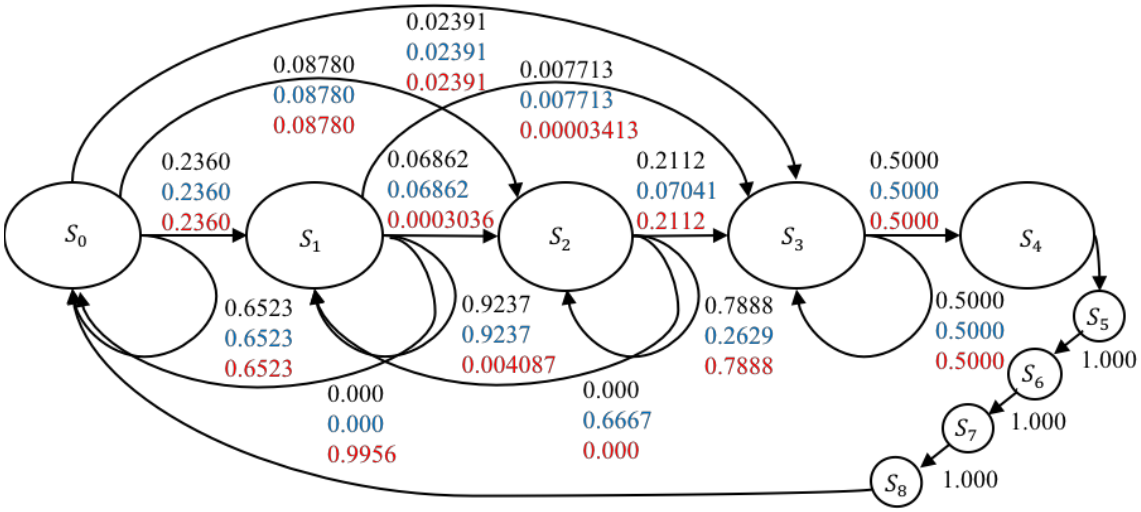

Figure 17. Annotated State Transition Diagram with State Transition Probabilities as Shown in Equations 31–33. Adapted from [6].

3. Limiting State Probabilities

For the POMDP model in this thesis, defensive actions are not always employed at the most effective time due to the defender observations not always being synonymous with actual state of the system. Limiting state probabilities allow us to determine the percentage of attacks that the defender could prevent if defenses were used optimally. Using Equation 2 with each of the transition matrices $P_{T,N}$, $P_{T,SR}$, and $P_{T,IR}$, we determined

the limiting state probabilities of the attack state, $\overline{p}_{4,N}$, $\overline{p}_{4,SR}$, and $\overline{p}_{4,IR}$ to be 0.0438, 0.0126, and 0.0459, respectively. The theoretical percentage of the attacks prevented with service reconfigurations is determined as

$$\alpha = \left(1 - \frac{\overline{p}_{4,SR}}{\overline{p}_{4,N}}\right) \times 100 \,. \tag{34}$$

Based on Equation 34, the system would suspend 71% of attacks if service reconfigurations were the only form of defense available and were employed optimally. Similarly, we determined that the system would suspend 0% of attacks if IP reconfigurations were the only form of defense available and were employed optimally. In summary, if service reconfigurations are employed effectively and there is no uncertainty surrounding the state in which the system lies, the system will prevent 71% of attacks. However, solely using IP reconfigurations as a defense method will not prevent any attacks within this POMDP model.

4. Observation Probabilities

The observation matrices of our POMDP model range from a model with full observability to three different variations of that model with partial observability, known as Case 1, Case 2, and Case 3. The 9×5 observability matrix or full observability matrix is given by

$$P_O = \begin{bmatrix} p_D & p_{MD} & p_{MD} & p_{MD} & p_{MD} \\ p_{MD} & p_D & p_{MD} & p_{MD} & p_{MD} \\ p_{MD} & p_{MD} & p_D & p_{MD} & p_{MD} \\ p_{MD} & p_{MD} & p_{MD} & p_D & p_{MD} \\ p_{MD} & p_{MD} & p_{MD} & p_{MD} & p_D \\ p_{MD} & p_{MD} & p_{MD} & p_{MD} & p_D \\ p_{MD} & p_{MD} & p_{MD} & p_{MD} & p_D \\ p_{MD} & p_{MD} & p_{MD} & p_{MD} & p_D \\ p_{MD} & p_{MD} & p_{MD} & p_{MD} & p_D \end{bmatrix} . \tag{35}$$

where the probability of detection (p_D) is 1 and the probability of miss detection (p_{MD}) is 0. When p_D = 1 and p_{MD} = 0, the defender is always certain of the true state of the system.

Three cases were considered to model a potential real-world scenario of detecting an attack. In these cases, the defender is never certain of the true state of the system in our POMDP model; thus, p_D is always less than 1. All three of the cases were studied with p_D ranging from 0.95 to 0.55 in decrements of 0.10. We set the lower limit of p_D at 0.55 because at 0.50 there is no difference between using moving target defense with an optimal policy and using a periodic defense. We denote the observation matrices for Case 1, Case 2, and Case 3, as $P_{O,1}$, $P_{O,2}$, $P_{O,3}$, respectively, and they are given by

$$P_{O,1} = \begin{bmatrix} p_D & \frac{p_{MD}}{4} & \frac{p_{MD}}{4} & \frac{p_{MD}}{4} & \frac{p_{MD}}{4} \\ \frac{p_{MD}}{4} & p_D & \frac{p_{MD}}{4} & \frac{p_{MD}}{4} & \frac{p_{MD}}{4} \\ \frac{p_{MD}}{4} & \frac{p_{MD}}{4} & p_D & \frac{p_{MD}}{4} & \frac{p_{MD}}{4} \\ \frac{p_{MD}}{4} & \frac{p_{MD}}{4} & \frac{p_{MD}}{4} & p_D & \frac{p_{MD}}{4} \\ \frac{p_{MD}}{4} & \frac{p_{MD}}{4} & \frac{p_{MD}}{4} & \frac{p_{MD}}{4} & p_D \\ \frac{p_{MD}}{4} & \frac{p_{MD}}{4} & \frac{p_{MD}}{4} & \frac{p_{MD}}{4} & p_D \\ \frac{p_{MD}}{4} & \frac{p_{MD}}{4} & \frac{p_{MD}}{4} & \frac{p_{MD}}{4} & p_D \\ \frac{p_{MD}}{4} & \frac{p_{MD}}{4} & \frac{p_{MD}}{4} & \frac{p_{MD}}{4} & p_D \\ \frac{p_{MD}}{4} & \frac{p_{MD}}{4} & \frac{p_{MD}}{4} & \frac{p_{MD}}{4} & p_D \end{bmatrix}, \tag{36}$$

$$P_{O,2} = \begin{bmatrix} p_D & p_{MD} & 0 & 0 & 0 \\ p_{MD} & p_D & 0 & 0 & 0 \\ p_{MD} & 0 & p_D & 0 & 0 \\ p_{MD} & 0 & 0 & p_D & 0 \\ p_{MD} & 0 & 0 & 0 & p_D \\ p_{MD} & 0 & 0 & 0 & p_D \\ p_{MD} & 0 & 0 & 0 & p_D \\ p_{MD} & 0 & 0 & 0 & p_D \\ p_{MD} & 0 & 0 & 0 & p_D \end{bmatrix}, \tag{37}$$

$$P_{O,3} = \begin{bmatrix} p_D & p_{MD} & 0 & 0 & 0 \\ \frac{p_{MD}}{2} & p_D & \frac{p_{MD}}{2} & 0 & 0 \\ 0 & \frac{p_{MD}}{2} & p_D & \frac{p_{MD}}{2} & 0 \\ 0 & 0 & \frac{p_{MD}}{2} & p_D & \frac{p_{MD}}{2} \\ 0 & 0 & 0 & p_{MD} & p_D \\ 0 & 0 & 0 & p_{MD} & p_D \\ 0 & 0 & 0 & p_{MD} & p_D \\ 0 & 0 & 0 & p_{MD} & p_D \\ 0 & 0 & 0 & p_{MD} & p_D \end{bmatrix}. \quad (38)$$

We refer to $P_{O,1}$ as the "neighbor" case because p_{MD} lies in the neighboring states of p_D with the probabilities in all other states equal to 0. In this case, if the defender incorrectly observes the state it is in, it will observe a state that is neighboring the true state of the system. This error presents a best-case scenario for partial observability. By observing a state that is neighboring the true state, the defender is likely to employ defenses that would be effective in the true state. For example, if the defender observes it is in state S_4 when the true state is actually S_3, even though the observation is incorrect, the defender is still likely to employ a vulnerability scan that would be effective in S_3. In $P_{O,1}$, even if the observation of the state is not synonymous with the true state of the system, there is still a chance at defending against a possible imminent attack.

We refer to $P_{O,2}$ as the "miss detection" case because p_{MD} always lies in S_0. In this case, if the defender incorrectly observes the state it is in, it will observe the start-state. When the defender believes it is in the start-state, it believes it is not being attacked to any degree and is not likely to employ any type of defense. For example, if the defender observes it is in S_0 when the true state is actually S_3, it is not likely to conduct a reconfiguration and the attack will be likely to progress forward. In $P_{O,2}$, if the observation of the state is not synonymous with the true state of the system, the defender will completely miss detect the imminent attack and will not be able to defend against it.

We refer to $P_{O,3}$ as the "combination" case because it is similar to a combination of $P_{O,1}$ and $P_{O,2}$. In this case, p_{MD} is divided equally amongst all remaining states and could lie in a neighboring state of p_D or in S_0. If the defender incorrectly observes the state it is

in, it is equally likely to observe any other state. In $P_{O,3}$, if the observation of the state is not synonymous with the true state of the system, there is both the possibility of completely miss detecting an imminent attack and the possibility of detecting an attack and defending against it.

5. Optimal Policy

The optimal policy for the POMDP model with full observability was determined using the Markov Decision Process (MDP) Toolbox in MATLAB (see Appendix B for the MATLAB code used in this thesis) [22]. The optimal policy was found to be [N, N, SR, N, N] corresponding to states $[S_0, S_1, S_2, S_3, S_4]$, respectively, where the symbols N, SR, and IR represent the actions nil (N), service reconfiguration (SR), and IP reconfiguration (IR), respectively. This policy shows that it is optimal for the system to conduct a service reconfiguration in the vulnerability state and refrain from conducting any reconfigurations in all other states.

In the optimal policy, only service reconfigurations are employed. Although IP reconfigurations were available, they were never selected because a POMDP-based approach tailors optimization efforts to defend against the specific attack faced. In the case of attacks presented in our POMDP model, the IP reconfiguration does not offer enough return on investment to be included in the optimal policy. This was expected given the assessment of limiting state probabilities considered in Section IV.A.3. Given that an IP reconfiguration is more expensive than a service reconfiguration and that it does not reduce the time spent in the attack state, it is not useful in our efforts towards optimizing MTD techniques.

B. SIMULATION AND RESULTS

The simulation phase of this thesis involved determining the DESPOT parameters under which our POMDP model performed best, solving the POMDP with full observability to verify that it followed the optimal policy, and solving Cases 1, 2, and 3. In this section, we discuss the simulation phase as well as the hardware that was used to run these simulations, how we processed the data returned by DESPOT, and how we analyzed

that data in a way that led us to determine if the moving target defense techniques within the models were optimized.

1. Solving with DESPOT

Each time we ran DESPOT to solve a POMDP, we manually modified the following parameters within the algorithm to meet our desired specifications: the number of decisions, particles, search depth, and time per decision. We used a total of 320 CPUs divided among 10 Dell R420 servers. Each of the CPUs had between 4 and 8GB RAM, and each of the servers belonged to the Intel Xeon family.

2. Post-processing

Post-processing involved converting the data that DESPOT returned from each run into a format that we are able to analyze in MATLAB. We created a parser in PYTHON for this data conversion. DESPOT outputs a standard collection of data for each decision epoch that occurs. The parser we created transformed that data into an array format. We included the following data in each array: round, step, action, state, observation, and reward. For example, if we chose 40,000 decisions for one run with DESPOT, after parsing, we would end up with 40,000 arrays.

3. Data Analysis

Data analysis involved developing equations and figures to interpret our parsed data. We developed three equations to determine the percentage of attacks that were prevented, the total cost to the system in seconds, and the total network availability. We also created histograms depicting the distributions of actions according to state to determine the majority policy for our POMDP model with full observability and for Cases 1, 2 and 3.

The estimated percentage of attacks prevented $\hat{\alpha}$ is determined as

$$\hat{\alpha} = \left(1 - \frac{\left(\frac{\xi_a}{\xi_d}\right)}{\overline{\overline{p}}_{4,N}} \right) \times 100, \tag{39}$$

where ξ_a is the total number of attacks and ξ_d is the total number of decision epochs. The total number of attacks include only S_4, as $S_5 \rightarrow S_8$ are all part of the recovery delay states that transition the model back to the start state. Equation 39 is similar to Equation 34; however, in Equation 39, we estimated $\overline{p}_{4,SR}$ using the approximation $\overline{p}_{4,SR} \cong \frac{\xi_a}{\xi_d}$.

The total cost η describes the normalized sum of the cost of defenses employed to prevent attacks, and it was determined as

$$\eta = \frac{(\xi_{SR} \times \eta_{SR}) + (\xi_{IR} \times \eta_{IR})}{\xi_d}, \tag{40}$$

where ξ_{SR} is the total number of service reconfigurations, ξ_{IR} is the total number of IP reconfigurations, η_{SR} is the cost of a service reconfiguration, η_{IR} is the cost of an IP reconfiguration, and ξ_d is the total number of decision epochs. Cost was determined by adding up the total number of service and IP reconfigurations and multiplying by their respective costs, and it was normalized by dividing by the total number of decision epochs.

The percentage availability ν describes the percentage of time the network services were available to the user, and it was determined as

$$\nu = \frac{\left(\left(\frac{\zeta - \eta_{SR}}{\zeta}\right) \times \xi_{SR}\right) + \left(\left(\frac{\zeta - \eta_{IR}}{\zeta}\right) \times \xi_{IR}\right) + (\xi_N)}{\xi_{SR} + \xi_{IR} + \xi_N} \times 100, \tag{41}$$

where ζ represents the maximum network availability in between decision epochs, ξ_{SR} is the total number of service reconfigurations, ξ_{IR} is the total number of IP reconfigurations, η_{SR} is the cost of a service reconfiguration, η_{IR} is the cost of an IP reconfiguration, and ξ_N is the number of nil reconfigurations. In Equation 41, the costs of reconfigurations are subtracted from ζ. We chose a value for ζ to convey that, in our POMDP model, reconfigurations cannot be triggered faster than they can be executed and that attacks have a severe impact on the system. We used $\zeta = 10$ because it is the first whole number larger than the costs of both types of reconfigurations and because it is significantly smaller than the cost of an attack. In our POMDP model, although reconfigurations decrease the overall

network availability, they are likely to be employed because the cost of reconfiguring is much less than that of being attacked.

4. Determining DESPOT Parameters

After solving the POMDP and determining the optimal policy and limiting state probabilities, initial simulations were conducted with full observability to determine if DESPOT default parameters would produce the best results. In these simulations, we used a range of values for the three DESPOT parameters discussed in Table 1: particles, search depth, and decision time. These parameters were of interest because they were most likely to affect the quality of decisions being made during the simulations.

The default parameters for particles, search depth, and decision time are 500, 90, and 1, respectively. We tested four categories of parameters: default parameters, reduced particles and depth, reduced particles, and reduced depth. We also tested each of these categories with different decision times ranging from 1 to 5 seconds in increments of 1. The parameters for each of the categories are shown in Table 3.

Table 3. Categories of DESPOT Parameters Tested

	Particles	Search Depth	Time Per Decision (sec)	Number of Decisions
Default Parameters	500	90	1,2,3,4,5	40,000
Reduced Particles and Depth	50	10	1,2,3,4,5	40,000
Reduced Particles	50	90	1,2,3,4,5	40,000
Reduced Depth	500	10	1,2,3,4,5	40,000

The results of testing each category of parameters described in Table 3 are shown in Figures 18 through 20. The results were analyzed in terms of percentage attacks, absolute value of cost, and percentage availability. The analysis of percentage attacks versus decision time in Figure 18 does not show a significant difference between any of the categories, as the attack percentages remained fairly constant around 70%. The analysis of the absolute value of the cost versus decision time in Figure 19 shows that the lowest cost is achieved with default parameters, followed by reduced depth, then by reduced particles, and lastly by reduced particles and depth. The analysis of percentage availability versus decision time in Figure 20 shows that the highest availability is achieved with default parameters, once again followed by reduced depth, then by reduced particles, and lastly by reduced particles and depth. The same trend is observed for both the cost and availability analysis showing that default parameters perform best with our simulations by allowing the highest quality decision making.

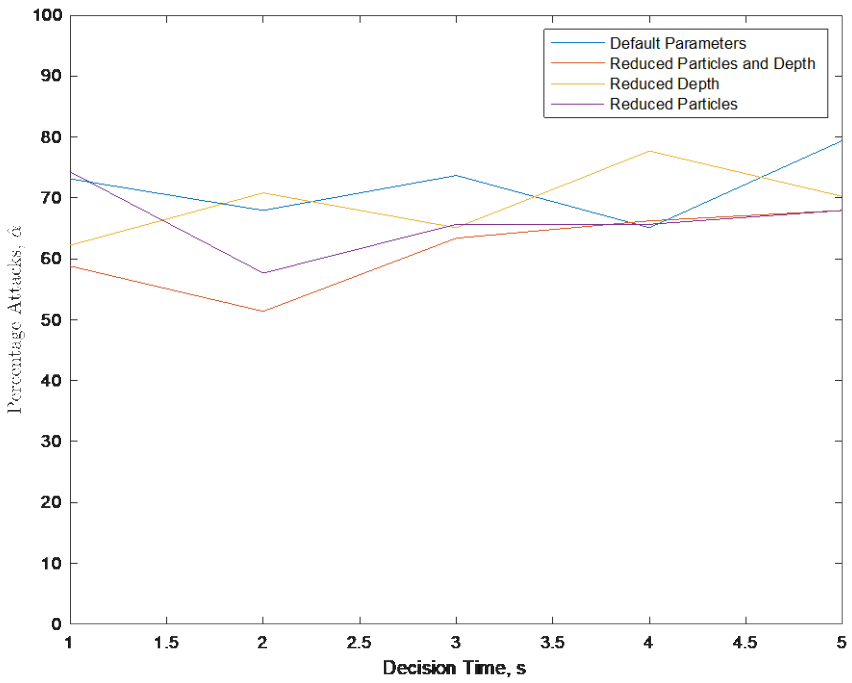

Figure 18. Comparison of Percentage Attacks as a Function of Decision Time for a Range of DESPOT Parameter Values as Specified in Table 3

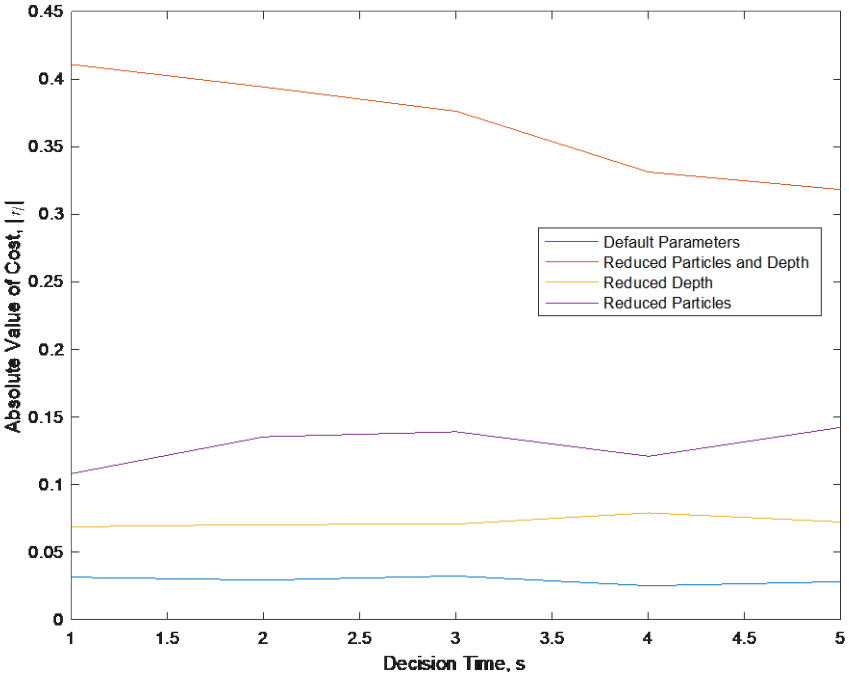

Figure 19. Comparison of Absolute Value of Cost as a Function of Decision Time for a Range of DESPOT Parameter Values as Specified in Table 3

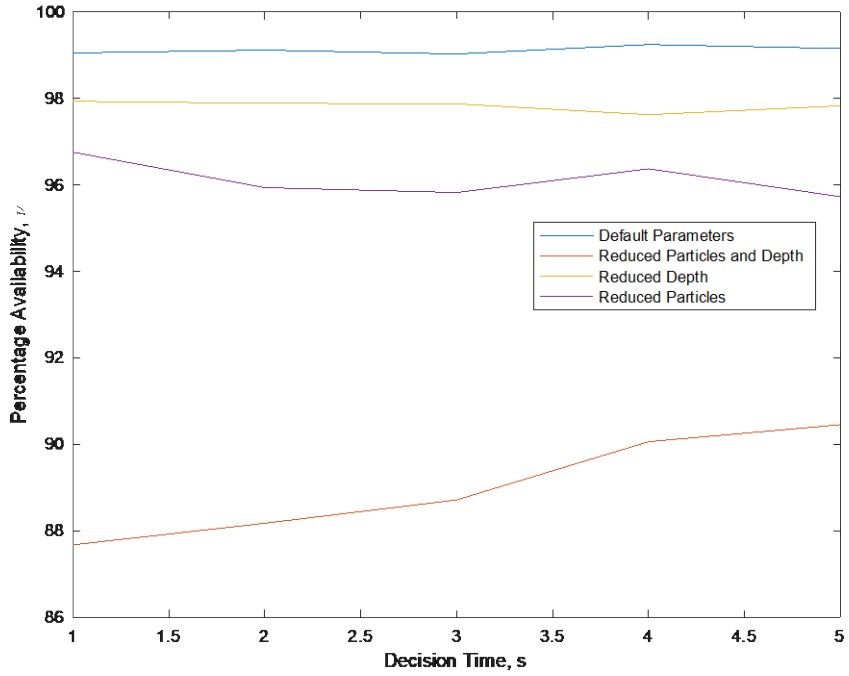

Figure 20. Comparison of Percentage Availability as a Function of Decision Time for a Range of DESPOT Parameter Values as Specified in Table 3

5. Verification of Optimal Policy

Once we determined that default DESPOT parameters performed best with our simulations, we used DESPOT to solve the POMDP with full observability using 40,000 decision epochs. In order to determine the policy that decision making followed during this run, we obtained a normalized histogram representing the estimated probabilities of the actions that occurred in each state within the set of 40,000 decision epochs. From these histograms, we were able to determine the action that occurred most frequently in each state, which we refer to collectively as the majority policy. These histograms are depicted in Figure 21 in order of $S_0 \rightarrow S_4$, from left to right. The data in Figure 21 is summarized in Table 4. This data shows that the majority policy for this data is [N, N, SR, N, N], which is synonymous with the optimal policy. This data also shows that IP reconfigurations never occurred in any of the states, which was expected based on the analysis of the optimal policies in Section IV.A.5. This test was used as a control in this thesis because it proved that DESPOT was able to optimize moving target defense techniques by following the optimal policy before uncertainty was added to the POMDP model.

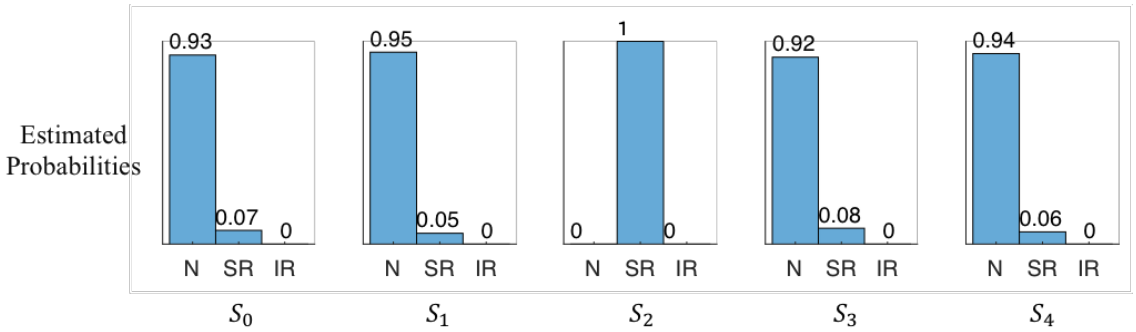

Figure 21. Estimated Probabilities of Actions by State for the POMDP Model with Full Observability (i.e., $p_D = 1$)

Table 4. Summary of the Relative Probabilities of Each Action in Figure 21 and the Majority Policy

p_D	States														
	S_0			S_1			S_2			S_3			S_4		
	Actions			Actions			Actions			Actions			Actions		
	N	SR	IR	N	SR	IR	N	SR	IR	N	SR	IR	N	SR	IR
1.00	0.93	0.07	0	0.95	0.05	0	0	1.00	0	0.92	0.08	0	0.94	0.06	0
	Majority Policy: [N, N, SR, N, N]														

6. Testing Uncertainty Models

Once the optimal policy was verified for the POMDP model with full observability, we used DESPOT to solve Cases 1, 2, and 3 using 40,000 decisions each. We solved each case with p_D ranging from 0.95 to 0.55 in decrements of 0.10. We analyzed the data in terms of attack percentage, cost, availability, and majority policy to understand how the addition of uncertainty affected decision making.

a. Attack percentage, Cost, and Availability Analysis

Plots for percentage attacks, absolute value of cost, and percentage availability versus p_D for Cases 1, 2, and 3 are shown in Figures 22 through 24. The percentage attacks analysis in Figure 22 showed that even as p_D decreased from 0.95 to 0.55, the percentage of attacks prevented remained around 70%. The cost analysis in Figure 23 showed that as p_D decreased, the absolute value of the cost increased from about 0.04 at the lowest point to about 0.18 at the highest point. The percentage availability analysis in Figure 24 showed that as p_D decreased, the availability increased from about 94.5% at the lowest point to 98.7% at the highest point.

In all three cases, as p_D decreased, the percentage of attacks remained about the same, the absolute value of the cost increased, and the percentage availability decreased. These trends show that as p_D decreased, the defender was less certain of the true state of the system requiring it to reconfigure more often and not always at the most effective time. A steady trend for attack percentage around 70% indicates that although the defender reconfigured more often, the increase in cost was due to more reconfigurations occurring

and not more attacks occurring; therefore, the increase in the defender reconfigurations was effective at preventing attacks at the expense of increased cost and decreased availability. Moreover, the defender was able to continue to prevent a similar percentage of attacks as the theoretical percentage calculated for service reconfigurations using Equation 34. It is notable that Case 2, the 'miss detection' case, performed the best out of all three cases with the lowest cost and highest availability. Additionally, for each of the three cases, there is a sharp increase in absolute value of the cost and sharp decrease in availability around p_D = 0.75 indicating that DESPOT becomes less effective at optimizing MTD techniques within our POMDP model at this point.

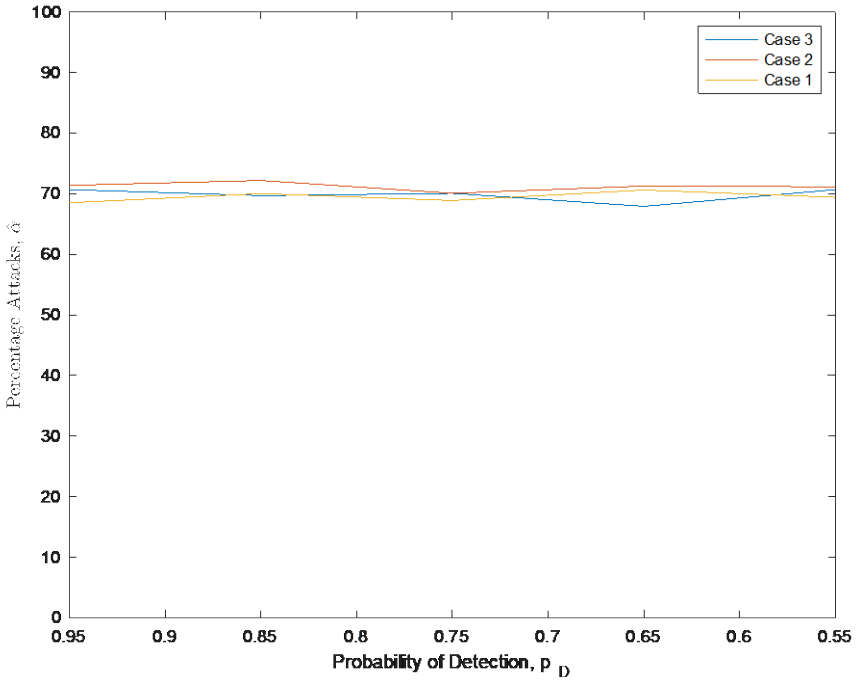

Figure 22. Comparison of Percentage Attacks as a Function of p_D for Cases 1, 2, and 3 as Specified in Equations 36, 37, and 38

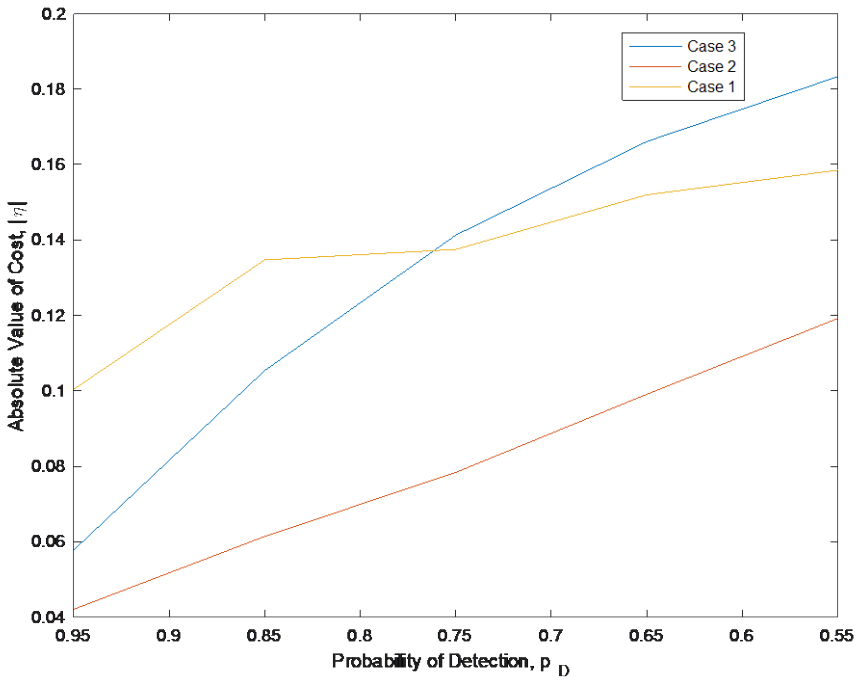

Figure 23. Comparison of Absolute Value of Cost as a Function of p_D for Cases 1, 2, and 3 as Specified in Equations 36, 37, and 38

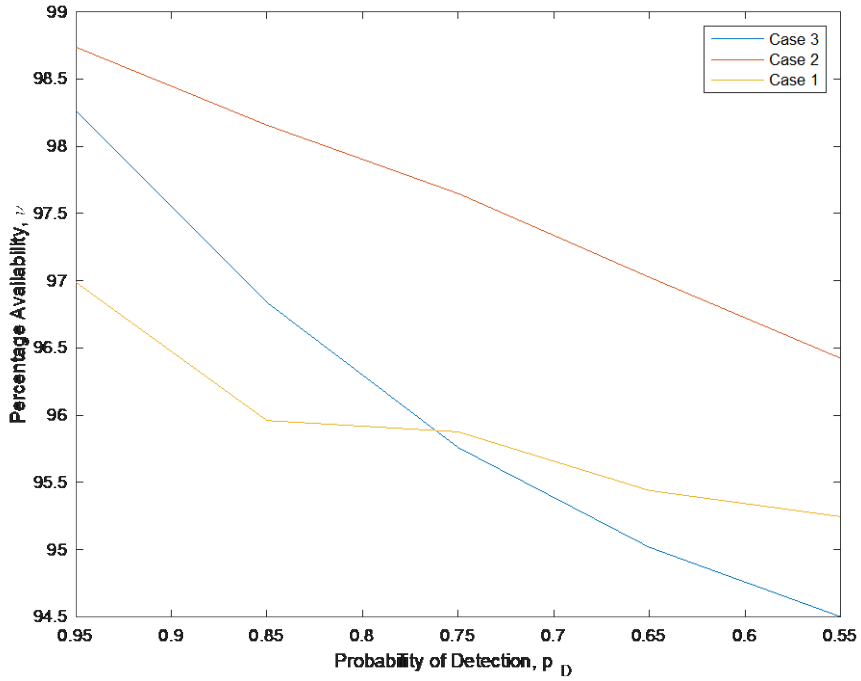

Figure 24. Comparison of Percentage Availability as a Function of p_D for Cases 1, 2, and 3 as Specified in Equations 36, 37, and 38

b. Policy Analysis

In order to determine the policy that decision making followed for each p_D tested for Cases 1, 2, and 3, we estimated the probabilities of the actions (i.e. normalized histogram values) that occurred in each state within each set of 40,000 decision epochs similar to that of Figure 21. A summary of the estimated probabilities for each action as well as the majority policy for each p_D tested for Cases 1, 2, and 3 is depicted in Tables 5, 6, and 7. For each case and each p_D level, the majority policy aligns with one of three types of policies: the optimal policy, a periodic policy, or a policy falling somewhere in between the optimal policy and a periodic policy. In our POMDP model, the optimal policy is represented by [N, N, SR, N, N], whereas a periodic policy is represented by [SR, SR, SR, SR, SR]. In the optimal policy, the defender conducts a reconfiguration in the S_2 state only; however, in a periodic policy, the defender disregards the optimal policy and conducts reconfigurations at fixed intervals due to uncertainty surrounding the true state of the system. Results for Case 1 in Table 5 show that the majority of decisions were made following the optimal policy when $p_D = 0.95$. However, as the uncertainty level increased (i.e., as p_D decreased), decision making began to resemble that of a periodic policy as service reconfigurations began to make up the majority of actions in states other than just S_2. Results for Case 2 in Table 6 followed the same pattern as Case 1; however, the majority policy trended towards a periodic policy at a slower rate in Case 2 than Case 1. Results for Case 3 in Table 7 followed the same trend as those of Cases 1 and 2; however, the policy trended towards a periodic policy the fastest in this case. In Case 3, the policy became periodic at $p_D = 0.75$; whereas, the policy never became fully periodic in Cases 1 and 2.

Table 5. Summary of the Estimated Probabilities of Actions and the Majority Policies for Each p_D for Case 1

p_D	States														
	S_0			S_1			S_2			S_3			S_4		
	Actions			Actions			Actions			Actions			Actions		
	N	SR	IR	N	SR	IR	N	SR	IR	N	SR	IR	N	SR	IR
0.95	0.8	0.2	0	0.53	0.47	0	0.03	0.97	0	0.38	0.62	0	0.84	0.16	0
	Majority Policy: [N, N, SR, SR, N]														
0.85	0.63	0.37	0	0.34	0.66	0	0.03	0.97	0	0.33	0.67	0	0.77	0.23	0
	Majority Policy: [N, SR, SR, SR, N]														
0.75	0.71	0.29	0	0.31	0.69	0	0.04	0.96	0	0.3	0.7	0	0.8	0.2	0
	Majority Policy: [N, SR, SR, SR, N]														
0.65	0.59	0.41	0	0.24	0.76	0	0.03	0.97	0	0.32	0.68	0	0.71	0.29	0
	Majority Policy: [N, SR, SR, SR, N]														
0.55	0.47	0.53	0	0.21	0.79	0	0.03	0.97	0	0.33	0.67	0	0.67	0.33	0
	Majority Policy: [SR, SR, SR, SR, N]														

Table 6. Summary of the Estimated Probabilities of Actions and Majority Policies for Each p_D for Case 2

p_D	States														
	S_0			S_1			S_2			S_3			S_4		
	Actions			Actions			Actions			Actions			Actions		
	N	SR	IR	N	SR	IR	N	SR	IR	N	SR	IR	N	SR	IR
0.95	0.6	0.4	0	0.88	0.12	0	0.01	0.99	0	0.87	0.13	0	0.94	0.06	0
	Majority Policy: [N, N, SR, N, N]														
0.85	0.37	0.63	0	0.79	0.21	0	0.02	0.98	0	0.87	0.13	0	0.9	0.1	0
	Majority Policy: [SR, N, SR, N, N]														
0.75	0.39	0.61	0	0.68	0.32	0	0.01	0.99	0	0.79	0.21	0	0.87	0.13	0
	Majority Policy: [SR, N, SR, N, N]														
0.65	0.41	0.59	0	0.56	0.44	0	0.01	0.99	0	0.74	0.26	0	0.84	0.16	0
	Majority Policy: [SR, N, SR, N, N]														
0.55	0.43	0.57	0	0.44	0.56	0	0.01	0.99	0	0.62	0.38	0	0.8	0.2	0
	Majority Policy: [SR, SR, SR, N, N]														

Table 7. Summary of the Estimated Probabilities of Actions and Majority Policies for Each p_D for Case 3

p_D	States														
	S_0			S_1			S_2			S_3			S_4		
	Actions			Actions			Actions			Actions			Actions		
	N	SR	IR	N	SR	IR	N	SR	IR	N	SR	IR	N	SR	IR
0.95	0.77	0.23	0	0.79	0.21	0	0.02	0.98	0	0.43	0.57	0	0.82	0.18	0
	Majority Policy: [N, N, SR, SR, N]														
0.85	0.53	0.47	0	0.54	0.46	0	0.04	0.96	0	0.26	0.74	0	0.48	0.52	0
	Majority Policy: [N, N, SR, SR, SR]														
0.75	0.35	0.65	0	0.35	0.65	0	0.04	0.96	0	0.15	0.85	0	0.3	0.7	0
	Majority Policy: [SR, SR, SR, SR, SR]														
0.65	0.24	0.76	0	0.23	0.77	0	0.04	0.96	0	0.08	0.92	0	0.13	0.87	0
	Majority Policy: [SR, SR, SR, SR, SR]														
0.55	0.14	0.86	0	0.14	0.86	0	0.04	0.96	0	0.07	0.93	0	0.1	0.9	0
	Majority Policy: [SR, SR, SR, SR, SR]														

Although the results from this analysis show that the majority policy varies for each case and each p_D, there is a common trend with all three. At lower uncertainty levels, the optimal policy, or a policy close to it, is followed, and at higher uncertainty levels, the policy becomes more and more periodic. It is also notable that IP reconfigurations never occurred in trials of the system. While this was expected, given the analysis of optimal policy analysis at $p_D = 1.0$ discussed in Section IV.A.5, it serves as additional validation that the system can achieve optimized MTD as p_D degrades. Taking all three cases into account, at $p_D = 0.75$, we can no longer trust the system to follow the optimal policy, and it is no longer advantageous to use DESPOT to solve the POMDP.

C. SUMMARY

The methodology introduced in Chapter 3 was implemented in this chapter in two sections. The first section explains the POMDP model in detail to include how we defined each aspect of our POMDP model as well as how we determined its optimal policy and limiting state probabilities. The second section explains the simulation results to include how we solved each of our POMDP cases with DESPOT and the results we obtained from the solutions. Consequently, the results show that the majority of decisions in our simulations were made following the optimal policy with no uncertainty. As uncertainty increased in Cases 1, 2, and 3, percentage attacks remained almost constant, the absolute value of cost increased, percentage availability decreased, and the majority policy became more and more periodic. In each analysis, it was clear that when p_D became 0.75, DESPOT did not perform as well at optimizing the MTD techniques within our POMDP model or at following the optimal policy. Therefore, we encourage the use of DESPOT in a cyber-defense scenario similar to our POMDP model as long as p_D is greater than 0.85.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSION

In this thesis, we developed a method for optimizing MTD techniques that involved developing a model, choosing a solution method, analyzing the results, and determining if optimization was achieved. We followed this method in order to reach our objective of optimizing MTD techniques by developing a POMDP model, solving different variations of it with DESPOT, and analyzing the results in terms of attack percentage, cost, availability percentage, and majority policy.

A. SIGNIFICANT CONTRIBUTIONS

The most significant contribution in this thesis is the development of a POMDP model as a framework to optimize MTD techniques and minimize cost. The majority policy of the POMDP with full observability was in line with the optimal policy validating that this framework is suitable as a platform for optimizing such techniques. Having a valid framework that can be modified for the purpose of optimizing cyber defense techniques is critical in a world of constantly evolving cyber threats.

Another contribution is determining the bounds on which DESPOT performs best as an MTD controller. The majority policies were closest to the optimal policy allowing the system to maximize system availability when $p_D > 0.85$. However, below 0.85, reconfigurations became periodic in nature causing percentage availability to decrease. Within the limitations of this research, these findings show that MTD techniques can be optimized using a POMDP and that DESPOT offers a valid solution above $p_D = 0.85$.

Our results suggest that performance gains are possible when defenses are tailored to attack context. Cyber defenses that seem advantageous in general may not be advantageous in the specific context of attacks faced. For example, given the attack model developed from [5], our POMDP-based policy includes only service reconfigurations even though IP reconfigurations were also available. In practice, tailoring defenses to the attacks faced prevents investment in defenses that seem useful at face value but fail to return effective defensive advantage.

B. RECOMMENDATIONS FOR FUTURE WORK

This thesis provides several avenues for future work. Considering the proposed scheme in Chapter III, any part of the scheme could be updated to improve the end goal of optimizing the implementation of MTD techniques in order to minimize cost and maximize availability for the defender. There is room for improvement in each of the steps of our method of approach including model development, model solution, and analysis of results; however, the majority of potential improvement lies in model development.

The data used to develop the POMDP model in this thesis offers deep insight into one specific attack-defense context. Because it is so tailored, conducting similar analysis across other attack models and with more defenses incorporated will offer similar deep insight into new models. This will facilitate broader and more generalized conclusions about what POMDP-based defense brings to the table. For example, there are almost certainly attack scenarios in which IP reconfigurations would offer return on investment.

The states of the POMDP model are also very specific to the attack defense context in this thesis. The attack model has recovery delay states that could potentially affect decision making when the system suspects it is near the attack state, S_4. There is a possibility the system will incur the cost associated with S_4 in order to get back to a more desirable state, such as the start state, S_0. Modifications made to the model states could mitigate this potential undesirable effect and facilitate the POMDP model in performing exactly as designed.

APPENDIX A. POMDP MODEL

This appendix contains the xml-formatted encoding of the POMDP model for ingest into DESPOT, built from the template available from [21].

```
<pomdpx                version="0.1"                id="autogenerated"
xsi:noNamespaceSchemaLocation="pomdpx.xsd">
  <Description>Cyber defense scenario
  </Description>
  <Discount>0.95</Discount>
  <Variable>
    <StateVar        vnamePrev="attack_0"        vnameCurr="attack_1"
fullyObs="false">
    <ValueEnum>start ts vs el x1 x2 x3 x4 x5</ValueEnum>
    </StateVar>
    <ObsVar vname="IDS">
    <ValueEnum>start ts vs el x</ValueEnum>
    </ObsVar>
    <ActionVar vname="defense">
    <ValueEnum>nil d1 d2</ValueEnum>
    </ActionVar>
    <RewardVar vname="reward"/>
  </Variable>
  <InitialStateBelief>
    <CondProb>
    <Var>attack_0</Var>
    <Parent>null</Parent>
    <Parameter type="TBL">
      <Entry>
        <Instance>-</Instance>
        <ProbTable>1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0</ProbTable>
      </Entry>
    </Parameter>
    </CondProb>
  </InitialStateBelief>
  <RewardFunction>
    <Func>
    <Var>reward</Var>
    <Parent>defense attack_1</Parent>
    <Parameter type="TBL">
      <Entry>
        <Instance>nil -</Instance>
        <ValueTable>0 0 0 0 -695 0 0 0 0</ValueTable>
      </Entry>
      <Entry>
        <Instance>d1 -</Instance>
        <ValueTable>-0.635 -0.635 -0.635 -0.635 -695.635 -0.635 -0.635
-0.635 -0.635</ValueTable>
      </Entry>
```

```
      <Entry>
        <Instance>d2 -</Instance>
        <ValueTable>-9.59 -9.59 -9.59 -9.59 -704.59 -9.59 -9.59 -9.59
-9.59</ValueTable>
      </Entry>
    </Parameter>
    </Func>
  </RewardFunction>
  <ObsFunction>
    <CondProb>
    <Var>IDS</Var>
    <Parent>attack_1</Parent>
    <Parameter type = “TBL”>
      <Entry>
        <Instance>start -</Instance>
        <ProbTable>1.0 0.0 0.0 0.0 0.0</ProbTable>
      </Entry>
      <Entry>
        <Instance>ts -</Instance>
        <ProbTable>0.0 1.0 0.0 0.0 0.0</ProbTable>
      </Entry>
      <Entry>
        <Instance>vs -</Instance>
        <ProbTable>0.0 0.0 1.0 0.0 0.0</ProbTable>
      </Entry>
      <Entry>
        <Instance>el  -</Instance>
        <ProbTable>0.0 0.0 0.0 1.0 0.0 </ProbTable>
      </Entry>

      <Entry>
        <Instance>x1 -</Instance>
        <ProbTable>0.0 0.0 0.0 0.0 1.0</ProbTable>
      </Entry>
      <Entry>
        <Instance>x2 -</Instance>
        <ProbTable>0.0 0.0 0.0 0.0 1.0</ProbTable>
      </Entry>
      <Entry>
        <Instance>x3 -</Instance>
        <ProbTable>0.0 0.0 0.0 0.0 1.0</ProbTable>
      </Entry>
      <Entry>
        <Instance>x4 -</Instance>
        <ProbTable>0.0 0.0 0.0 0.0 1.0</ProbTable>
      </Entry>
      <Entry>
        <Instance>x5 -</Instance>
        <ProbTable>0.0 0.0 0.0 0.0 1.0</ProbTable>
      </Entry>
    </Parameter>
    </CondProb>
  </ObsFunction>
```

```
  <StateTransitionFunction>
    <CondProb>
    <Var>attack_1</Var>
    <Parent>defense attack_0</Parent>
    <Parameter type="TBL">
      <Entry>
        <Instance>* start -</Instance>
        <ProbTable>0.6523  0.2360  0.08780  0.02391  0.0  0.0  0.0  0.0
0.0</ProbTable>
      </Entry>
      <Entry>
        <Instance>* x1 -</Instance>
        <ProbTable>0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0</ProbTable>
      </Entry>
       <Entry>
        <Instance>* x2 -</Instance>
        <ProbTable>0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0</ProbTable>
      </Entry>
      <Entry>
        <Instance>* x3 -</Instance>
        <ProbTable>0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0</ProbTable>
      </Entry>
      <Entry>
        <Instance>* x4 -</Instance>
        <ProbTable>0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0</ProbTable>
      </Entry>
      <Entry>
        <Instance>* x5 -</Instance>
        <ProbTable>1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0</ProbTable>
      </Entry>      <Entry>
        <Instance>* el -</Instance>
        <ProbTable>0.0 0.0 0.0 0.5 0.5 0.0 0.0 0.0 0.0</ProbTable>
      </Entry>
      <Entry>
        <Instance>nil ts -</Instance>
        <ProbTable>0.0  0.9237  0.06862  0.007713  0.0  0.0  0.0  0.0
0.0</ProbTable>
      </Entry>
      <Entry>
        <Instance>d1 ts -</Instance>
        <ProbTable>0.0  0.9237  0.06862  0.007713  0.0  0.0  0.0  0.0
0.0</ProbTable>
      </Entry>
      <Entry>
        <Instance>d2 ts -</Instance>
        <ProbTable>0.9956 0.004087 0.0003036 0.00003143 0.0 0.0 0.0
0.0 0.0</ProbTable>
      </Entry>
      <Entry>
        <Instance>nil vs -</Instance>
        <ProbTable>0.0   0.0   0.7888   0.2112   0.0   0.0   0.0   0.0
0.0</ProbTable>
      </Entry>
```

```
      <Entry>
        <Instance>d1 vs -</Instance>
        <ProbTable>0.0  0.6667  0.2629  0.07041  0.0  0.0  0.0  0.0
0.0</ProbTable>
      </Entry>
      <Entry>
        <Instance>d2 vs -</Instance>
        <ProbTable>0.0   0.0   0.7888   0.2112   0.0   0.0   0.0   0.0
0.0</ProbTable>
      </Entry>
    </Parameter>
    </CondProb>
   </StateTransitionFunction>
</pomdpx>
```

APPENDIX B. RESULTS INTERPRETATION SCRIPT

This appendix includes the functions written in MATLAB to analyze the results we received after solving each of our POMDPs with DESPOT. It includes the scripts for testing DESPOT default parameters, for determining the optimal policy and limiting state probabilities, for verifying the optimal policy, and for testing Cases 1, 2, and 3 with different uncertainty levels.

```
%% Despot default parameters

%Default Parameters
dectime9=[1 2 3 4 5];
attacks9=[.7311 .6796 .7368 .6510 .7941] ;
cost9=[-.0316 -.0295 -.0325 -.0252 -.0283];
avail9=[99.0523 99.1158 99.0253 99.2444 99.1523];

%Both Reduced
dectime10= [1 2 3 4 5];
attacks10=[.5881 .5137 .6339 .6625 .6796];
cost10= [-.4107 -.3941 -.3761 -.3312 -.3182];
avail10=[87.6788 88.1759 88.7166 90.0634 90.4545];

%Reduced depth
attacks11=[.6224 .7082 .6510 .7769 .7025];
cost11= [-.0689 -.0705 -.0709 -.0792 -.0724];
avail11=[97.9331 97.8854 97.8728 97.6234 97.8267];

%reduced particles
attacks12=[.7426 .5767 .6568 .6568 .6796];
cost12= [-.1082 -.1354 -.1392 -.1210 -.1424];
avail12=[96.7555 95.9357 95.8238 96.3709 95.7288];

figure(7)
plot(dectime9, attacks9)
title('Attacks vs. Decision Time')
xlabel('Decision Time(s)')
ylabel('Attacks')
hold on
plot(dectime9, attacks10)
hold on
plot(dectime9, attacks11)
hold on
plot(dectime9, attacks12)
ylim([0 1])
legend('Default Parameters','Reduced Particles and Depth','Reduced
Depth','Reduced Particles')

figure(8)
```

```
plot(dectime9, cost9)
title('Cost vs. Decision Time')
xlabel('Decision Time(s)')
set(gca, 'YDir','reverse')
ylabel('Cost')
hold on
plot(dectime9, cost10)
hold on
plot(dectime9, cost11)
hold on
plot(dectime9, cost12)
legend('Default Parameters','Reduced Particles and Depth','Reduced
Depth','Reduced Particles')

figure(9)
plot(dectime9, avail9)
title('Availability vs. Decision Time');
xlabel('Decision Time(s)')
ylabel('Availability')
hold on
plot(dectime9, avail10)
hold on
plot(dectime9, avail11)
hold on
plot(dectime9, avail12)
legend('Default Parameters','Reduced Particles and Depth','Reduced
Depth','Reduced Particles')

% Optimal Policy
%1: nil 2: d1: service 3: d2: ip
%markov decision process
T(:,:,1)=[0.6523 0.2360 0.08780 0.02391 0 0 0 0 0
  0 .9237 .06862 .007713 0 0 0 0 0
  0 0 .7888 .2112 0 0 0 0 0
  0 0 0 .5 .5 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
  1 0 0 0 0 0 0 0 0]
T(:,:,2)=[0.6523 0.2360 0.08780 0.02391 0 0 0 0 0
  0 .9237 .06862 .007713 0 0 0 0 0
  0 .6667 .2629 .07041 0 0 0 0 0
  0 0 0 .5 .5 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
  1 0 0 0 0 0 0 0 0]
T(:,:,3)=[0.6523 0.2360 0.08780 0.02391 0 0 0 0 0
  .9956 .004087 .0003036 .00003143 0 0 0 0 0
  0 0 .7888 .2112 0 0 0 0 0
  0 0 0 .5 .5 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
  1 0 0 0 0 0 0 0 0]

Reward= [0 0 0 0 -695 0 0 0 0
  -.635 -.635 -.635 -.635 -695.635 -.635 -.635 -.635 -.635
  -9.59 -9.59 -9.59 -9.59 -704.59 -9.59 -9.59 -9.59 -9.59]'

[V,OptPol,iter,cpu_time] = mdp_policy_iteration(T,Reward,.95);

% Limiting State Probabilities
T1=T(:,:,1)^41
T2=T(:,:,2)^41
T3=T(:,:,3)^41


% Verify Optimal Policy
data= readtable('Parsed_5_90_500_.txt');
action= data.Var3; % 0, 1, or 2
state= data.Var4;
STATE=categorical(state);

Z=categories(STATE);
cats1 = countcats(STATE);
attacks1 = 1-(((cats1(5)/length(action))*100)/4.38);

COST1=cost(data);
AVAILIBILITY1=availibil(data);
NEWREC1=newrec(data,1);

%% Case 1, Case 2, Case 3

Percent_all=[95 85 75 65 55];
% Case 3
R3_attack=[.7066 .6969 .7005 .6791 .7066];
R3_avail=[98.2657 96.8390 95.7596 95.0171 94.5020];
R3_cost=[-.0578 -.1054 -.1413 -.1661 -.1833];

% Case 2
R2_attack=[.7137 .7218 .7005 .7126 .7112];
R2_avail=[98.7362 98.1572 97.6489 97.0259 96.4233];
R2_cost=[-.0421 -.0614 -.0784 -.0991 -.1192];

% Case 1
R_attack=[.6851 .7002 .6888 .7061 .6939];
R_avail=[96.9886 95.9605 95.8763 95.4406 95.2461];
R_cost=[-.1004 -.1347 -.1375 -.1520 -.1585];

figure(4)
plot(Percent_all, R3_attack)
hold on
plot(Percent_all, R2_attack)
hold on
```

```
plot(Percent_all , R_attack)
xlabel('Probability of Detection, p_D');
ylabel('Percentage Attacks, $$\hat{\alpha}$$','Interpreter','Latex');
ylim([0 100])
legend('Case 3', 'Case 2', 'Case 1')
set(gca, 'XDir','reverse')
hold off

figure(5)
plot(Percent_all, abs(R3_cost))
hold on
plot(Percent_all, abs(R2_cost))
hold on
plot(Percent_all , abs(R_cost))
xlabel('Probability of Detection, p_D ');
ylabel('Absolute Value of Cost, |\eta|');
legend('Case 3', 'Case 2', 'Case 1')
set(gca, 'YDir')
set(gca, 'XDir','reverse')
hold off

figure(6)
plot(Percent_all, R3_avail)
hold on
plot(Percent_all, R2_avail)
hold on
plot(Percent_all , R_avail)
xlabel('Probability of Detection, p_D ');
ylabel('Percentage Availability, \nu');
legend('Case 3', 'Case 2', 'Case 1')
set(gca, 'XDir','reverse')
hold off

% Case 1, Case 2, Case 3
%% 95%

data= readtable('data');
action= data.Var3; % 0, 1, or 2
state= data.Var4;
STATE=categorical(state);
Z=categories(STATE);
cats1 = countcats(STATE);
attacks1 = 1-(((cats1(5)/length(action))*100)/4.38); % count x1
COST1=cost(data);
AVAILIBILITY1=availibil(data);
NEWREC1=newrec(data,1);

%% 85%

data2= readtable('data');
action2= data2.Var3;
state2= data2.Var4;
STATE2=categorical(state2);
cats2 = countcats(STATE2);
```

```
attacks2 = 1-(((cats2(5)/length(action2))*100)/4.38);
COST2=cost(data2);
AVAILIBILITY2=availibil(data2);
NEWREC2=newrec(data2,2);

%% 75%

data3= readtable('data');
action3= data3.Var3;
state3= data3.Var4;
STATE3=categorical(state3);
cats3 = countcats(STATE3);
attacks3 = 1-(((cats3(5)/length(action3))*100)/4.38);
COST3=cost(data3);
AVAILIBILITY3=availibil(data3);
NEWREC3=newrec(data3,3);

%% 65%

data4= readtable('data');
action4= data4.Var3;
state4= data4.Var4;
STATE4=categorical(state4);
cats4 = countcats(STATE4);
attacks4 = 1-(((cats4(5)/length(action4))*100)/4.38);
COST4=cost(data4);
AVAILIBILITY4=availibil(data4);
NEWREC4=newrec(data4,4);

%% 55%

data5= readtable('data');
action5= data5.Var3;
state5= data5.Var4;
STATE5=categorical(state5);
cats5 = countcats(STATE5);
attacks5 = 1-(((cats5(5)/length(action5))*100)/4.38);
COST5=cost(data5);
AVAILIBILITY5=availibil(data5);
NEWREC5=newrec(data5,5);

%% plot

G = categorical({'start','ts','vs','el','x'});
G = reordercats(G,{'start','ts','vs','el','x'});

figure(9)
subplot(2,4,1)
histogram(STATE,G,'Normalization','probability')
title('1s')
subplot(2,4,2)
histogram(STATE2,G,'Normalization','probability')
title('2s')
subplot(2,4,3)
```

```
histogram(STATE3,G,'Normalization','probability')
title('3s')
subplot(2,4,4)
histogram(STATE4,G,'Normalization','probability')
title('4s')
subplot(2,4,5)
histogram(STATE5,G,'Normalization','probability')
title('5s')

%% display

attacks_tot=[attacks1 attacks2 attacks3 attacks4 attacks5]
avail_tot=[AVAILIBILITY1 AVAILIBILITY2 AVAILIBILITY3 AVAILIBILITY4
AVAILIBILITY5]

%% Functions

%Function: reconfigurations by state
function [STATS]=stats(z)
data=z;
A=data.Var3;
[a b]=find(A==1);
B=categorical(data.Var4(a));
C=categorical(data.Var5(a));
categB=(categories(B));
categC=(categories(C));
catsB=countcats(B);
catsC=countcats(C);
catsB1=[catsB(1);catsB(2);catsB(3);catsB(4);sum(catsB(5:9))];
%output: how often did reconfigurations occur in each state
STATS=[catsB1./90330 catsC./90330];
end

%function to test when observation and state are equal
function [obs_state_tf]=testifequal(a,b)
data1=a;
data2=b;
tf=[];
for i=1:length(a)
  if strcmp(data1(i),data2(i))
    tf=[tf;1];
  else
    tf=[tf;0];
  end
end
tf1=tf;
obs_state_tf=sum(tf1==1);
end

% FUNCTION: Cost
function COST=cost(x)
data1=x;
data2=data1.Var3;
d=find(data2==1);
```

```
d1=find(data2==2);
d2=find(data2==0);
e=data2(d);
e1=data2(d1);
e2=data2(d2);
f=length(e);
f1=length(e1);
f2=length(e2);
datax=data1.Var4;
dx2=strfind(datax, 'x2');
dx3=strfind(datax, 'x3');
dx4=strfind(datax, 'x4');
dx5=strfind(datax, 'x5');
fx1=length(dx2);
fx2=length(dx3);
fx3=length(dx4);
fx4=length(dx5);
sum1=fx1+fx2+fx3+fx4
z=length(data2)-sum1;
COST=((f*.635)+(f1*9.59))/z;
end

% Function: Availability
function AVAILIBILITY=availibil(x)
data1=x;
data2=data1.Var3;
d=find(data2==1);
d1=find(data2==2);
d2=find(data2==0);
e=data2(d);
e1=data2(d1);
e2=data2(d2);
f=length(e);
f1=length(e1);
f2=length(e2);
AVAILIBILITY= (((((10-.635)/10)*f)+(((10-9.59)/10)*f1)+(((10-
0)/10)*f2))/(f+f1+f2))*100
end

% When is system reconfiguring; histograms
function NEWREC=newrec(x,y)
data=x
data2=data.Var4; %state
data3=categorical(data2);
data4=data.Var3;
data5= circshift(data4,-1); % shift rows up
d=find(data3=='start');
d1=find(data3=='ts');
d2=find(data3=='vs');
d3=find(data3=='el');
d4=find(data3=='x1'); % d4= d5,d6,d7,d8
D=data5(d);
D1=data5(d1);
D2=data5(d2);
D3=data5(d3);
```

```
D4=data5(d4);

figure(y)
subplot(1,5,1)
histogram(D,[-0.5,0.5,1.5,2.5],'Normalization','probability')
h=histogram(D,[-0.5,0.5,1.5,2.5],'Normalization','probability')
x = h.BinEdges ;
y1 = h.Values ;
text(x(1:end-
1),y1,num2str(round(y1',2)),'vert','bottom','horiz','left');
box off
xticklabels({'N','SR','IR'})
set(gca,'YTick', [])
%title('start')
subplot(1,5,2)
histogram(D1,[-0.5,0.5,1.5,2.5],'Normalization','probability')
h=histogram(D1,[-0.5,0.5,1.5,2.5],'Normalization','probability')
x = h.BinEdges ;
y1 = h.Values ;
text(x(1:end-
1),y1,num2str(round(y1',2)),'vert','bottom','horiz','left');
box off
xticklabels({'N','SR','IR'})
set(gca,'YTick', [])
%title('ts')
subplot(1,5,3)
histogram(D2,[-0.5,0.5,1.5,2.5],'Normalization','probability')
h=histogram(D2,[-0.5,0.5,1.5,2.5],'Normalization','probability')
x = h.BinEdges ;
y1 = h.Values ;
text(x(1:end-
1),y1,num2str(round(y1',2)),'vert','bottom','horiz','left');
box off
xticklabels({'N','SR','IR'})
set(gca,'YTick', [])
%title('vs')
subplot(1,5,4)
histogram(D3,[-0.5,0.5,1.5,2.5],'Normalization','probability')
h=histogram(D3,[-0.5,0.5,1.5,2.5],'Normalization','probability')
x = h.BinEdges ;
y1 = h.Values ;
text(x(1:end-
1),y1,num2str(round(y1',2)),'vert','bottom','horiz','left');
box off
xticklabels({'N','SR','IR'})
set(gca,'YTick', [])
%title('el')
subplot(1,5,5)
histogram(D4,[-0.5,0.5,1.5,2.5],'Normalization','probability')
h=histogram(D4,[-0.5,0.5,1.5,2.5],'Normalization','probability')
x = h.BinEdges ;
y1 = h.Values ;
text(x(1:end-
1),y1,num2str(round(y1',2)),'vert','bottom','horiz','left');
box off
```

```
xticklabels({'N','SR','IR'})
set(gca,'YTick', [])
%title('attack')

NEWREC= figure(y)
end

% Function: reconfiguration statistics
function RECONFIG=reconfig(m)
set1=m;
set2=set1.Var6;
[q r]=find(set2==-0.635);
[q1 r1]=find(set2==-9.59);
b=categorical(set1.Var5(q));
bb=categorical(set1.Var5(q1));
b1= num2cell((countcats(b)));
bb1=num2cell((countcats(bb)));
b2=categories(b);
B= cell2mat(b1) + cell2mat(bb1);
B= num2cell(B);
RECONFIG=[b1 bb1 b2];
end
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] Cybersecurity Ventures, "2019 official annual cybercrime report," Sausalito, CA, USA, Rep. CV-HG, 2019. [Online]. Available: https://www.herjavecgroup.com/wp-content/uploads/2018/12/CV-HG-2019-Official-Annual-Cybercrime-Report.pdf

[2] G. C. Wilshusen, "Agencies need to improve the implementation of federal approach to securing systems and protecting against intrusions," Washington, DC, USA, GAO Report No. GAO-19-105, 2018.

[3] X. Zhou, Y. Lu, Y. Wang, and X. Yan, "Overview on moving target defense," in *2018 IEEE 3rd Int. Conf. on Image, Vision and Computing (ICIVC)*, Chongqing, China, 2018. [Online]. doi: 10.1109/icivc.2018.8492800

[4] W. Connell, L. H. Pham, and S. Philip, "Analysis of concurrent moving target defenses," in *Proc. of the 5th ACM Workshop on Moving Target Defense*, Toronto, Canada, 2018. [Online]. doi: 10.1145/3268966.3268972

[5] 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 Int. Conf. on Dependable Systems and Networks (DSN'05)*, Yokohama, Japan, 2005. [Online]. doi: 10.1109/DSN.2005.18

[6] A. McAbee, "A moving target defense scheme with overhead control via partially observable Markov decision processes," presented in Electrical and Computer Engineering Department, Naval Postgraduate School, Monterey, CA, Apr. 22, 2020.

[7] 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," Lockheed Martin Corp., 2014. [Online]. Available: https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/LM-White-Paper-Intel-Driven-Defense.pdf

[8] B. C. Ward, S. R. Gomez, R. W. Skowyra, D. Bigelow, J. N. Martin, J. W. Landry, and H. Okhravi. "Survey of cyber moving targets second edition," Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, MA, USA, Rep. 1228, 2018. [Online]. Available: web.mit.edu/br26972/www/pubs/mt_survey.pdf

[9] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY, USA: Wiley, 1994.

[10] H. Kamalzadeh and M. Hahsler, “POMDP: Introduction to Partially Observable Markov Decision Processes,” The Comprehensive R Archive Network, December 8, 2019. [Online]. Available: https://cran.r-project.org/web/packages/pomdp/vignettes/POMDP.html

[11] A. R. Cassandra, L. P. Kaelbling, and M. L. Littman. “Acting optimally in partially observable stochastic domains,” Dept. of Comp. Sci., Brown University, Providence, RI, USA, Rep. CS-94-20, 1994. [Online]. doi: 10.5555/864411

[12] D. Braziunas, “POMDP solution methods,” Dept. of Comp. Sci, University of Toronto, 2003. [Online]. Available: semanticscholar.org/paper/POMDP-solution-methods-Braziunas/3cd3291a57831c5e80863fd5065cbe7c1f371d28

[13] N. Ye, A. Somani, D. Hsu, and W. S. Lee, “DESPOT: online POMDP planning with regularization,” *J. of Artif. Intell. Res.*, vol. 58, no. 1, pp. 231–266, Jan. 2017. [Online]. doi: 10.1613/jair.5328.

[14] N. Lord, “What is an advanced persistent threat? APT definition,” Digital Guardian, September 11, 2018. [Online]. Available: https://digitalguardian.com/blog/what-advanced-persistent-threat-apt-definition.

[15] P. K. Manadhata, D. K. Kaynar, and J. M. Wing. “A formal model for a system’s attack surface,” School of Comp. Sci., Carnegie Mellon University, Pittsburgh, PA, USA, Rep. CMU-CS-07-144, 2007. [Online]. Available: www.cs.cmu.edu/~wing/publications/CMU-CS-07-144.pdf

[16] C. W. Therrien and M. Tummala, *Probability and Random Processes for Electrical and Computer Engineers*, 2nd ed. Boca Raton, FL, USA: CRC Press, 2012.

[17] B. Givan and R. Parr, “An introduction to Markov decision processes,” presented at Dagstuhl Seminars, Germany, November, 2001. [Online]. Available: https://engineering.perdue.edu/~givan/talks/mdp-tutorial.pdf

[18] S. Guerreiro, “Decision-making in partially observable environments,” in *2014 IEEE 16th Conf. on Bus. Informatics*, Geneva, Switzerland, 2014, pp. 159–166. [Online]. doi: 10.1109/CBI.2014.15.

[19] C. Boutilier, T. Dean, and S. Hanks, “Decision-theoretic planning: structural assumptions and computational leverage,” *J. of Artif. Intell. Res.*, vol. 11, no. 1, pp. 1–94, Jul. 1999. [Online]. doi: 10.1613/jair.575.

[20] E. Miehling, M. Rasouli, and D. Teneketzis, “A POMDP approach to the dynamic defense of large-scale cyber networks,” *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 10, pp. 2490–2505, Oct. 2018. [Online]. doi: 10.1109/TIFS.2018.2819967.

[21] N. Ye, A. Somani, D. Hsu, and W. Lee. “Approximate POMDP planning online toolkit,” GitHub, accessed June 22, 2020. [Online]. Available: https://github.com/AdaCompNUS/despot

[22] M. Cros, “Markov Decision Processes (MDP) toolbox,” MATLAB, 2020. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/25786-markov-decision-processes-mdp-toolbox

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California