Optimal Timing of Moving Target Defense: A Stackelberg Game Model

Henger Li and Zizhan Zheng
Department of Computer Science, Tulane University, New Orleans, USA
Email: {hli30, zzheng3}@tulane.edu

***Abstract*—As an effective approach to thwarting advanced attacks, moving target defense (MTD) has been applied to various domains. Previous works on MTD, however, mainly focus on deciding the sequence of system configurations to be used and have largely ignored the equally important timing problem. Given that both the migration cost and attack time vary over system configurations, it is crucial to jointly optimize the spatial and temporal decisions in MTD to better protect the system from persistent threats. In this work, we propose a Stackelberg game model for MTD where the defender commits to a joint migration and timing strategy to cope with configuration-dependent migration cost and attack time distribution. The defender's problem is formulated as a semi-Markovian decision process and a nearly optimal MTD strategy is derived by exploiting the unique structure of the game.**

I. Introduction

Cyber-attacks are becoming increasingly more adaptive and sophisticated. One example is Advanced Persistent Threats (APTs) [1], an emerging class of continuous and stealthy hacking processes launched by incentive driven entities. To avoid immediate detection and obtain long-term benefit, an advanced attacker may carefully cover its tracks, e.g., by internationally operating in a "low-and-slow" fashion [2]. The stealth and persistent nature makes these attacks extremely difficult to defense using traditional techniques that focus on one-shot attacks of known types.

An important obstacle in combating stealthy attacks is *information asymmetry*. An advanced attack often involves an information collection stage (e.g., through probing the system) to dynamically identify the best target to attack. In contrast, a defender typically knows much less about a stealthy and adaptive attacker. To revert the information asymmetry, an promising approach is moving target defense (MTD), where the defender constantly updates the system configuration to increase the attacker's uncertainty. By exploiting the diversity and randomness at different system layers, various MTD techniques have been proposed including dynamic networks [3], [4], dynamic platforms [5], dynamic runtime environments [6], dynamic software [7], and dynamic data [8].

In addition to empirical evaluation of domain specific MTD techniques, decision and game theoretic approaches have recently been adopted to derive more cost-effective MTD solutions. In particular, a zero-sum dynamic game for MTD is proposed in [9] where a fixed migration cost (i.e., the cost of switching from one configuration to another) is assumed. More recently, Bayesian Stackelberg games (BSGs) have been applied to MTD [10], where the defender commits to an *i.i.d.* strategy independent of the real-time system configuration, leading to a suboptimal strategy. In our previous work [11], we have proposed a Markovian modeling of MTD where the decision on the next system configuration to be used depends on the current one and derived the optimal Stackelberg strategy.

An important limitation of existing game models for MTD, however, is that the *temporal* decision has been largely ignored. Previous studies mainly focus on the *spatial* decision, i.e., what is the next configuration to be used, while assuming a simplified decision on timing. In particular, constant attack times and periodic migration policies are commonly assumed in previous work. However, different system configurations typically require different techniques and expertise to set up and to identify and exploit vulnerability. Thus, both the migration cost and the amount of time that the attacker needs to take down a system are configuration dependent. Both of them should be taken into consideration when deciding *when* to move as large attack times (or migration costs) imply less frequent updates. Therefore, a simple periodic migration strategy is far from satisfactory.

In this paper, we make the first effort on the joint optimization of spatial and temporal decisions in MTD. We extend our Markovian modeling of MTD in [11] by introducing attack times that are both random and configuration dependent. We consider a Stackelberg game model with the defender as the leader and the attacker as the follower. Stackelberg games provide a natural framework for studying information asymmetry and have been broadly applied in cybersecurity [10], [12], [13]. In our setting, the defender commits to a stationary MTD strategy at the beginning of the game, which includes both a configuration transition matrix (spatial decision) and a set of defense periods, one for each configuration (temporal decision). Instead of minimizing the long-term discounted cost as in [11], we consider the more challenging time-average cost objective in this work, which is more reasonable for patient attackers targeting long-term advantages. The problem of finding the best strategy for the defender is formulated as a semi-Markov decision process (SMDP) [14] with continuous decision variables. Although SMDPs with continuous decisions are difficult to solve in general, we show that the classic value iteration (VI) algorithm can be applied to our problem to obtain a nearly optimal stationary strategy. We further derive an efficient solution to the Min-Max problem in each iteration of VI by utilizing the unique structure of the MTD game.

We have made the following contributions in this paper.

- We propose a new active defense paradigm that incorporates *spatial* and *temporal* decisions to achieve robust moving target defense.
- We extend the Bayesian Stackelberg game (BSG) model

by considering Markovian defense strategies, which are more general than the repeated decisions in BSG and are more appropriate for MTD.
- We derive a nearly optimal defense strategy based on the value iteration technique and propose an efficient algorithm for each iteration by utilizing the unique structure of the Min-Max problem in the MTD game.

The rest of the paper is organized as follows. We review the related work on MTD games in Section II and present the game model and problem formulation in Section III. The optimal defense strategy and its analysis are discussed in Section IV. We evaluate our solution in Section V and conclude the paper in Section VI.

II. Related Work

Several game theoretic models have been proposed for MTD in the last few years [15]. A zero-sum dynamic game for MTD is proposed in [9], where each player chooses its action independently in each round according to a mixed strategy and gets immediate feedback on its payoff. However, the zero-sum assumption does not hold in many security scenarios. Further, a fixed migration cost is assumed in [9], which neglects the heterogeneity in configurations. More recently, Bayesian Stackelberg games (BSGs) have been applied to MTD in web applications [10], where the defender commits to an $i.i.d.$ migration strategy, which is suboptimal when the migration cost is configuration dependent. A BSG model for the closely related cyber deception problem is studied in [12]. Several Markov models for MTD have also been proposed recently [16], [17]. However, these works focus on analyzing the expected time needed to compromise a system under simple defense strategies instead of deriving optimal MTD strategies. In our recent work [11], we have extended the BSG models by introducing Markovian strategies into MTD while still considering periodic migrations as in previous works. Initiated by the FlipIt game [18], optimal timing of security updates has received a lot of interest recently [19]–[21], where instead of switching between configurations, the system is recovered after a certain time period. However, these studies do not apply to the optimal timing of MTD directly.

III. MTD Game Model and Problem Formulation

In this section, we present our game theoretic model for MTD and formulate the defender's optimization problem. Figure 1 gives an example of our attack-defense model.

A. System Model

System Configurations: We consider a system to be protected and two players, an attacker and a defender. The system has a set of configuration parameters that the defender can choose from. Examples include IP addresses, network topology, OS versions, memory address space layout, etc. To meet the system's integrity and performance requirement, only a subset of configurations is valid, which is defined as the system configuration space, denoted by S. Let $n = |S|$ denote the number of configurations.

Defense Model: The defender constantly migrates the system configuration to increase the attacker's uncertainty. We assume that a migration happens instantaneously subject to a cost m_{ij} if the system moves from configuration i to configuration j. We allow $m_{ii} > 0$ to model the cost of recovering the system to the same configuration. Let M denote the matrix of migration costs $\{m_{ij}\}_{n\times n}$. A continuous time horizon is considered. Let t_k denote the time instance when the k-th migration happens and s_k the system configuration in the k-th defense period (from t_{k-1} to t_k). At the end of the k-th defense period, the defender picks the next configuration s_{k+1} with probability $p_{s_k s_{k+1}}$. We assume $t_0 = 0$ and let s_0 denote the initial configuration (before t_0).

S	set of system configurations
n	number of configurations
a_j	random attack time for configuration j
p_{ij}	transition probability from configuration i to j
P	transition probability matrix
$\mathbf{p}_i$	the i-th row in P
α	lower bound of p_{ij}
m_{ij}	migration cost from configuration i to j
M	migration cost matrix
τ_i	length of the defense period when the previous configuration is i
$\overline{\tau}$, $\underline{\tau}$	maximum/minimum defense period
γ	a parameter in transforming SMDP to MDP
δ	step size in searching for τ in Algorithm 1
ω	a parameter controlling the stopping criterion in Algorithm 1

Table I: List of symbols in the paper

We assume that the defender adopts a *stationary* strategy consisting of (1) a transition matrix $P = \{p_{ij}\}_{n\times n}$ where p_{ij} is the probability of moving to configuration j when the system is currently in configuration i, and (2) a vector $\{\tau_i\}_{i\in S}$, where τ_i is the next defense period to be used if the system is currently in configuration i. According to this definition, the $k-th$ defense period only depends on s_{k-1} but not s_k (see Figure 1 for an example). This is to simplify the decision problem as we discuss below. We may also consider strategies where τ_k depends on s_k only or both s_{k-1} and s_k, which is left to our future work. Without loss of generality, we assume that $\tau_i \in [\underline{\tau}, \overline{\tau}]$ for any i where $\underline{\tau} > 0$ and $\overline{\tau} < \infty$. Let $\mathbf{p}_i$ denote the i-th row of P.

Attack Model: We consider a persistent attacker that continuously probes and attacks the system. We assume that once a migration happens, the attacker learns this fact immediately and makes a guess on the new configuration. Further, the amount of time needed to compromise the system under configuration j is modeled as a random variable a_j with distribution A_j and is $i.i.d.$ across attacks. Consider the k-th defense period. Let $\hat{s}_k$ denote the attacker's guess of s_k. Under the stationary defense strategy described above, the probability that the attacker's guess is correct is $\Pr(\hat{s}_k = s_k) = p_{s_{k-1}\hat{s}_k}$. The expected amount of time that the system is compromised in the k-th period then becomes $p_{s_{k-1}\hat{s}_k}\mathbb{E}[\max(\tau_{s_{k-1}} - a_{\hat{s}_k}, 0)]$ where the expectation is with respect to the randomness of attack time.

Stackelberg Game: We assume that the attacker always learns s_k at the end of the k-th defense period (a worst-case scenario from the defender's perspective). Consequently, the attacker

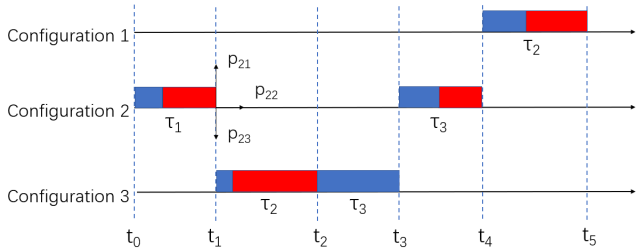


Figure 1: An example of the game model where configuration 1 is the initial configuration. A blue (resp. red) block denotes a time interval when the system is protected (resp. compromised). τ_i is the length of the current defense period when the previous configuration is i.

may also learn the defender's stationary strategy once enough samples are collected. To simplify the analysis, we assume that the defender announces its strategy at the beginning of the game. We further assume that the attacker is myopic and always exploits the most beneficial configuration according to the defender's strategy and the previous system configuration it observed. We then have $\hat{s}_k = \text{argmax}_{j\in S} p_{s_{k-1}j}\mathbb{E}[\max(\tau_{s_{k-1}} - a_j, 0)]$. Effectively, we consider a Stackelberg game with the defender as the leader and the attacker as the follower. As is typical in security games, we assume that the defender knows the attack time distribution (but not its realization). It is important to note that our game model is more general than the Bayesian Stackelberg Game (BSG) models in [10], [12] since the leader (defender) commits to a configuration-dependent Markovian strategy rather than a simple $i.i.d.$ strategy as in BSG where p_{ij} is a constant across i. To simplify the notation, we let $w_{ij} = \mathbb{E}[\max(\tau_i - a_j, 0)]$ for $i, j \in S$.

B. Defender's Problem as an SMDP

The defender's objective is to strike a balance between the loss from attacks and the cost of migration. To this end, we formulate the defender's problem as an average-cost semi-Markov decision process (SMDP) as follows. We define the state of the system as the set of configurations S. Let s_0 be the initial state (before the game begins). We consider stationary policies only. Each time the system is in state i, a control $\mu(i) \triangleq (\mathbf{p}_i, \tau_i)$ is applied, the defender then incurs an expected cost $c(i, \mu(i)) = \max_j(p_{ij}w_{ij}) + \sum_j p_{ij}m_{ij}$, and the system moves to state j with probability p_{ij}. Note that the cost function includes both the expected loss from attacks as well as the expected migration cost. For a given policy μ, the time-average cost of the defender starting from an initial state s_0 is defined as:

$$C_\mu(s_0) = \limsup_{N\to\infty} \frac{\sum_{k=0}^{N-1} c(s_k, \mu(s_k))}{\sum_{k=0}^{N-1} \tau_{s_k}} = \limsup_{N\to\infty} \frac{\sum_{k=0}^{N-1} [\max_j(p_{s_kj}w_{s_kj}) + \sum_j p_{s_kj}m_{s_kj}]}{\sum_{k=0}^{N-1} \tau_{s_k}} \quad (1)$$

The defender's goal is to commit to a policy μ that minimizes its time-average cost for any initial state. We thus obtain an infinite-horizon SMDP with average cost criterion and continuous decision variables.

IV. Optimal MTD Strategies

In this section, we propose efficient algorithms to find a nearly optimal solution to the defender's problem.

A. Approximation and Transformation

The SMDP defined in (1) has a finite state space S and a compact action space $[0, 1]^n \times [\underline{\tau}, \overline{\tau}]$. There are two major challenges to solve the SMDP. First, when an arbitrary transition matrix P is allowed, the Markov chain associated with a given stationary policy is not necessarily unichain. Consequently, the time-average cost may vary over the initial configurations [14]. Second, the continuous decision variables make it challenging to apply standard techniques such as value iteration and policy iteration as the Min-Max problem (defined below) in each iteration can be difficult to solve.

We discuss how to address the second challenge in the next subsection. To address the first challenge, we impose the following constraint on P by requiring that $p_{ij} \geq \alpha$ for any i, j where $\alpha > 0$ is a small number. With this simple constraint, the unichain requirement is always satisfied. To understand why the assumption is reasonable, consider two stationary policies with transition matrices P and P', respectively, both of which are unichain. If $|P_{ij} - P'_{ij}| \leq \alpha$ for any i, j, then the stationary distribution of P is close to that of P' by making α small enough [22]. Thus, the loss of optimality is negligible for small enough α if we only consider unichain policies. On the other hand, in a multichain policy, some configurations are never used for MTD, which is unlikely to happen in practice as it reduces the attacker's uncertainty. A rigorous understanding of the multichain case is left to our future work.

With the above assumption and the fact that the single stage cost $c(i, \mu(i))$ is continuous in p_{ij} and τ_i, it is known that the optimal time-average cost is independent of the initial configuration, and further, there is a stationary deterministic policy that is optimal [14]. Moreover, we can apply a standard trick to transform the SMDP to a discrete-time MDP with average cost criterion defined below:

$$\tilde{C}_\mu(s_0) = \limsup_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} \tilde{c}(s_k, \mu(s_k)) \quad (2)$$

with the single stage cost and transition probabilities given by [14]:

$$\tilde{c}(i, \mu(i)) = \frac{\max_j (w_{ij}p_{ij}) + \sum_j p_{ij}m_{ij}}{\tau_i} \quad (3)$$

$$\tilde{p}_{ij} = \gamma \frac{p_{ij} - \delta_{ij}}{\tau_i} + \delta_{ij} \quad (4)$$

where $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ otherwise, and γ satisfies $0 < \gamma < \tau_i/(1 - p_{ii})$ for any $i \in S$ and $p_{ii} < 1$. We choose $\gamma = \underline{\tau}$ in this work. The original SMDP and the transformed discrete-time MDP have the same class of stationary policies. Further, for each stationary policy μ, $C_\mu(i) = \tilde{C}_\mu(i)$ for any $i \in S$. This result does not require any assumption about the chain structures of the Markov chains associated with the stationary policies.

Algorithm 1 Value Iteration algorithm for the MTD game

Input: $S, \underline{\tau}, \overline{\tau}, M, \alpha, \gamma, \epsilon, \delta$.
Output: τ^*, P^*.

1: $t = 0, V^0(i) = 0, \forall i \in S$;
2: **repeat**
3: $t = t + 1$;
4: **for** $i \in S$ **do**
5: $v = \infty$;
6: **for** $\tau = \underline{\tau}; \tau \leq \overline{\tau}; \tau = \tau + \delta$ **do**
7: $\mathbf{p}' = \arg\min_{\mathbf{p}} V^t(i, \mathbf{p}, \tau)$;
8: $v' = V^t(i, \mathbf{p}', \tau)$
9: **if** $v' < v$ **then**
10: $\mathbf{p}_i^* = \mathbf{p}', \tau_i^* = \tau, v = v'$;
11: $V^t(i) = v$;
12: $\overline{V} = \max_{i \in S} |V^t(i) - V^{t-1}(i)|$,
13: $\underline{V} = \min_{i \in S} |V^t(i) - V^{t-1}(i)|$;
14: **until** $\overline{V} - \underline{V} < \omega \underline{V}$

B. Value Iteration Algorithm

Given the transformation defined above, the problem then boils down to solving the average cost MDP in (2). Since $\tilde{p}_{ij} > 0$ for any i, j, the MDP is still unichain. As the state space is finite and the action space is a separable metric space, it is known that the standard value iteration algorithm converges to the optimal average cost [23]. The main challenge is to design an efficient solution to the Min-Max problem in each iteration as discussed below.

The VI algorithm (see Algorithm 1) maintains a value vector $V^t \in \mathbb{R}^{+^n}$ in each iteration t. Initially, $V^0(i) = 0$ for each $i \in S$. In iteration t, the algorithm solves the following Min-Max problem for each configuration $i \in S$ (lines 4-11):

$$\begin{aligned} V^t(i) &= \min_{\mathbf{p}_i, \tau_i} \Big[\tilde{c}(i, \mu(i)) + \sum_{j \in S} \tilde{p}_{ij} V^{t-1}(j) \Big] \\ &= \min_{\mathbf{p}_i, \tau_i} \Bigg[\frac{\max_j (w_{ij} p_{ij}) + \sum_j p_{ij}(m_{ij} + \gamma V^{t-1}(j))}{\tau_i} \\ &\qquad + (1 - \frac{\gamma}{\tau_i}) V^{t-1}(i) \Bigg] \\ &\quad s.t. \quad \mathbf{p}_i \in [\alpha, 1]^n, \sum_j p_{ij} = 1, \tau_i \in [\underline{\tau}, \overline{\tau}]. \end{aligned} \tag{5}$$

Let $V^t(i, \mathbf{p}, \tau)$ denote the value of the objective function in (5) when $\mathbf{p}_i = \mathbf{p}$ and $\tau_i = \tau$. The Min-Max problem is difficult to solve due to the coupling of P and τ. To this end,l dleretize the search space for τ_i. For each $\tau_i \in \{\underline{\tau}, \underline{\tau} + \delta, \underline{\tau} + 2\delta, ..., \bar{\tau}]$ where δ is a parameter, (5) is solved to search for the best $\mathbf{p}_i$ (lines 6-10). An efficient solution for this step is discussed below. A smaller δ gives a better solution at the expense of a higher searching overhead.

Algorithm 1 stops when $V^t(i) - V^{t-1}(i)$ is close to a constant across i (see lines 15-17 where ω is a parameter). When the algorithm stops, $V^t(i) - V^{t-1}(i)$ provides a good approximation of the optimal time-average cost. The error bound of the value iteration algorithm is established in [23].

C. Solving the Min-Max Problem

A major obstacle in implementing Algorithm 1 is to find an efficient solution to the Min-Max problem (5) for a fixed τ (line 7 in Algorithm 1). To this end, we first show that the optimal $\mathbf{p}$ to this problem has a simple structure, which significantly simplifies the problem.

In the following discussion, we consider the Min-Max problem for configuration i in iteration t and for a fixed τ. We drop the indices i and t to ease the notation. Let $m_j = m_{ij}, p_j = p_{ij}, w_j = w_{ij}$, and $V(j) = V^{t-1}(j)$. Let $w_{\min} = \min_{i \in S} w_i$, $w_{\max} = \max_{i \in S} w_i$, and $\rho = \frac{w_{\max}}{w_{\min}}$. Since the denominator in the first term and the second term in (5) are both are constants, it suffices to consider the following problem:

$$\begin{aligned} \min_{\mathbf{p}} \Big[\max_j (w_j p_j) + \sum_j p_j (m_j + \gamma V(j)) \Big] \\ s.t. \quad \mathbf{p} \in [\alpha, 1]^n, \sum_j p_j = 1. \end{aligned} \tag{6}$$

Let $U(\mathbf{p})$ denote the value of the objective function in (6) for a given $\mathbf{p}$. Let $\theta_j = m_j + \gamma V(j)$ denote the coefficient of p_j in the second term of (6). For a given $\mathbf{p}$, let k be any configuration with $w_k p_k = \max_{j \in S}(w_j p_j)$. We partition $S \backslash \{k\}$ into two sets where $A = \{a \in S : \theta_a > w_k + \theta_k\}$ and $B = S \backslash (A \cup \{k\})$. Let $\{b_j\}_{1 \leq j \leq |B|}$ denote the sequence of elements in B sorted in θ non-decreasingly.

Proposition 1. *For any optimal* $\mathbf{p}$ *to* (6)*,* $p_a = \alpha, \forall a \in A$.

Proof. Assume $p_a = \alpha + \epsilon$ for some $a \in A$ and $\epsilon > 0$. We construct a new solution $\mathbf{p}'$ with $p'_a = \alpha$, $p'_k = p_k + \epsilon$, and $p'_j = p_j$ for any other j. Observe that $\mathbf{p}'$ is a feasible solution and $k = \arg\max_j (w_j p'_j)$. It follows that $U(\mathbf{p}') - U(\mathbf{p}) = w_k(p'_k - p_k) + (p'_a - p_a)\theta_a + (p'_k - p_k)\theta_k = (w_k + \theta_k)\epsilon - \theta_a \epsilon < 0$ since $\theta_a > w_k + \theta_k$ for any $a \in A$. This contradicts the fact that $\mathbf{p}$ is an optimal solution. □

Proposition 2. *Assume* $\alpha \leq \frac{1}{n\rho}$*. There is an optimal* $\mathbf{p}$ *to* (6) *where we can find an index* $q \in \{1, 2, ..., |B|\}$ *such that* $p_{b_j} = \frac{w_k}{w_{b_j}} p_k$ *for* $1 \leq j \leq q$ *and* $p_{b_j} = \alpha$ *for* $q < j \leq |B|$.

Proof. We first make the following observation. Consider any optimal solution $\mathbf{p}$ to (6). Assume that there are $j_1, j_2 \in \{1, ...|B|\}$ such that $j_1 < j_2$, $p_{b_{j_1}} < \frac{w_k}{w_{b_{j_1}}} p_k$, and $p_{b_{j_2}} > \alpha$. We claim that we can construct a new optimal solution $\mathbf{p}'$ such that either $p'_{b_{j_1}} = \frac{w_k}{w_{b_{j_1}}} p'_k$ or $p'_{b_{j_2}} = \alpha$ (or both) while keeping other probabilities unchanged. To see this, let $\epsilon_1 = \frac{w_k}{w_{b_{j_1}}} p_k - p_{b_{j_1}}$ and $\epsilon_2 = p_{b_{j_2}} - \alpha$. We distinguish two cases.

Case 1: $\epsilon_1 \leq \epsilon_2$**:** we define $p'_{b_{j_1}} = p_{b_{j_1}} + \epsilon_1$, $p'_{b_{j_2}} = p_{b_{j_2}} - \epsilon_1$, and $p'_j = p_j$ for any other j. Observe that $\mathbf{p}'$ is a feasible solution and $p'_{b_{j_1}} = \frac{w_k}{w_{b_{j_1}}} p'_k$. Further, we still have $w_k p'_k = \max_j (w_j p'_j)$. It follows that $U(\mathbf{p}') - U(\mathbf{p}) = (\theta_{b_{j_1}} - \theta_{b_{j_2}})\epsilon_1 \leq 0$ since $j_1 < j_2$. Thus, $\mathbf{p}'$ is optimal.

Case 2: $\epsilon_1 > \epsilon_2$**:** we define $p'_{b_{j_1}} = p_{b_{j_1}} + \epsilon_2$, $p'_{b_{j_2}} = p_{b_{j_2}} - \epsilon_2$, and $p'_j = p_j$ for any other j. $\mathbf{p}'$ is again feasible and $p'_{b_{j_2}} = \alpha$, and we still have $w_k p'_k = \max_j (w_j p'_j)$. It follows that $U(\mathbf{p}') - U(\mathbf{p}) = (\theta_{b_{j_1}} - \theta_{b_{j_2}})\epsilon_2 \leq 0$ since $j_1 < j_2$. Thus, $\mathbf{p}'$ is optimal.

Algorithm 2 Solving the Min-Max problem (6)

Input: $S, \{w_i\}, \{m_i\}, \gamma, V, \alpha$.
Output: $\mathbf{p}^*$.

1: $\theta_j = m_j + \gamma V(j), \forall j \in S$;
2: $u = \infty$
3: **for** $k \in S$ **do**
4: $p_j = \alpha$, for all j such that $\theta_j > w_k + \theta_k$;
5: $B = \{b : \theta_b \leq w_k + \theta_k\}$;
6: $\{b_j\}$ = the sequence of items in B sorted in θ non-decreasingly;
7: **for** $q = 1; q \leq |B|; q = q + 1$ **do**
8: $p_k = \frac{1-|A|\alpha-(|B|-q+1)\alpha}{\sum_{j<q}\frac{w_k}{w_j}+1}$;
9: $p_{b_j} = \frac{w_k}{w_{b_j}} p_k$, $\forall j \leq q$, $p_{b_j} = \alpha$, $\forall j > q$;
10: **if** $w_k p_k + \sum_{j \in S} p_j \theta_j < u$ **then**
11: $u = w_k p_k + \sum_{j \in S} p_j \theta_j$;
12: $\mathbf{p}^* = \mathbf{p}$;

From the above observation, starting from any optimal solution $\mathbf{p}$, we can construct a new optimal solution $\mathbf{p}'$ in which there is an index $q \in \{1, 2, ..., |B|\}$ such that $p'_{b_j} = \frac{w_k}{w_{b_j}} p'_k$ for $1 \leq j < q$, $p_{b_j} = \alpha$ for $q < j \leq |B|$, and $p'_{b_q} \in [\alpha, \frac{w_k}{w_{b_q}} p'_k]$. We then show that, starting from such a $\mathbf{p}'$, we can construct a new optimal solution $\mathbf{p}''$ that satisfies the statement in the theorem. The main idea is to move a small amount of value from $\{p'_{b_j}, j < q\} \cup \{p'_k\}$ to p'_{b_q} or the other way around depending on which direction is more beneficial. To this end, we again distinguish two cases:

Case 1: $(1 + \sum_{j<q} \frac{w_k}{w_{b_j}})\theta_{b_q} > \sum_{j<q} \frac{w_k}{w_{b_j}} \theta_{b_j} + (w_k + \theta_k)$: Let $\epsilon > 0$ be a small value to be determined. We construct a new solution $\mathbf{p}''$ where $p''_k = p'_k + \epsilon$, $p''_{b_j} = p'_{b_j} + \frac{w_k}{w_{b_j}}\epsilon$ for all $j < q$, $p''_{b_q} = p'_{b_q} - \epsilon - \sum_{j<q} \frac{w_k}{w_{b_j}}\epsilon$, and $p''_{b_j} = p'_{b_j}$ for other j. Note that $p''_{b_j} = \frac{w_k}{w_{b_j}} p''_k$ is maintained for $j < q$. Further, we can choose ϵ so that $p''_{b_q} = \alpha$. It is easy to see that $\mathbf{p}''$ is a feasible solution. Further, $U(\mathbf{p}'') - U(\mathbf{p}') = \Big(-(1 + \sum_{j<q} \frac{w_k}{w_{b_j}})\theta_{b_q} + \sum_{j<q} \frac{w_k}{w_{b_j}} \theta_{b_j} + (w_k + \theta_k) \Big)\epsilon \leq 0$. Hence, $\mathbf{p}''$ is also optimal.

Case 2: $(1 + \sum_{j<q} \frac{w_k}{w_{b_j}})\theta_{b_q} \leq \sum_{j<q} \frac{w_k}{w_{b_j}} \theta_{b_j} + (w_k + \theta_k)$: We construct a new solution $\mathbf{p}''$ where $p''_k = p'_k - \epsilon$, $p''_{b_j} = p_{b_j} - \frac{w_k}{w_{b_j}}\epsilon$ for all $j < q$, $p''_{b_q} = p'_{b_q} + \epsilon + \sum_{j<q} \frac{w_k}{w_{b_j}}\epsilon$, and $p''_{b_j} = p'_{b_j}$ for other j. We again have $p''_{b_j} = \frac{w_k}{w_{b_j}} p''_k$ for $j < q$, and we can choose ϵ so that $p''_{b_q} = \frac{w_k}{w_{b_q}} p''_k$. We claim that when $\alpha \leq \frac{1}{n\rho}$, we further have (i) $p''_{b_j} \geq \alpha$ for $j \leq q$ and (ii) $p''_j w_j \leq w_k p''_k$ for $j \in A \cup \{b_{q+1}, b_{q+2}, ..., b_{|B|}\}$. From these properties, we can conclude that $\mathbf{p}''$ is a feasible solution, and $U(\mathbf{p}'') - U(\mathbf{p}') = \Big((1 + \sum_{j<q} \frac{w_k}{w_{b_j}})\theta_{b_q} - \sum_{j<q} \frac{w_k}{w_{b_j}} \theta_{b_j} - (w_k + \theta_k) \Big)\epsilon \leq 0$. Thus, $\mathbf{p}''$ is also optimal. The detailed proofs of the two claims can be found in our online technical report [24]. □

Based on the two propositions above, we then design an efficient solution to (6) (see Algorithm 2). The algorithm iterates over all $k \in S$. For a given k, two sets A and B are identified and $p_j = \alpha$ for $j \in A$ (line 4). We then search for a proper index $q \leq |B|$ and set the value of p_{b_j} for $b_j \in B$ according to Proposition 2 (lines 7-9). The running time of Algorithm 2 is dominated by sorting all the configurations according to their θ values for each k. Thus, the complexity of the algorithm is $O(n^2 \log n)$, which is much faster than searching the whole probability space.

V. Numerical Results

In this section, we evaluate our MTD strategy through numerical studies under different system settings and demonstrate its advantage by comparing it with two heuristic strategies where a fixed defense period is used for all configurations:

1) Random sampling (RS): The defender stays in the current configuration for a fixed duration τ and then moves to a new configuration with probability $1/n$. The optimal τ is obtained by solving the following problem.

$$\min_{\tau} \frac{\max_j \mathbb{E}(\max(\tau - a_j, 0)) + \frac{1}{n}\sum_{i,j} m_{ij}}{n\tau}$$

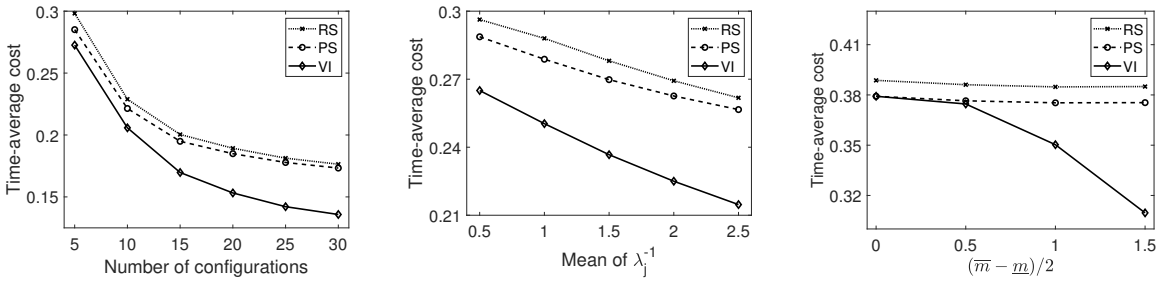
2) Proportional sampling (PS): The defender stays in the current configuration for a fixed duration τ and then moves to a new configuration j with probability p_j that is proportional to $w_j = \mathbb{E}[\max(\tau - a_j, 0)]$. The defense strategy $(\tau, \{p_j\})$ is obtained by solving the following problem.

$$\begin{aligned} \min_{\tau, \{p_j\}} \quad & \frac{\max_j (w_j p_j) + \sum_{i,j} p_i p_j m_{ij}}{\tau} \\ s.t. \quad & w_j = \mathbb{E}(\max(\tau - a_j, 0), \forall j \in S \\ & w_i p_i = w_j p_j, \forall i, j \in S \\ & \sum_{j \in S} p_j = 1 \end{aligned}$$

Simulation setup: In the simulations, the number of configurations n is chosen from $\{5, 10, ..., 30\}$. The set of migration costs are $i.i.d.$ samples from a uniform distribution. For each configuration j, its attack time a_j follows an exponential distribution with parameter λ_j, where λ_j is sampled from a uniform distribution and is $i.i.d.$ across j. In each simulation, we conduct 100 trials by taking 10 samples of the migration cost matrix M and 10 samples of $\{\lambda_j\}$. We set $\alpha = 0.01$ and $\omega = 0.01$ in Algorithms 1 and 2. We set $\underline{\tau} = 0.1, \bar{\tau} = 5$, and $\delta = 0.1$. For each $\tau_i \in \{0.1, 0.2, ..., 5\}$ and each λ_j, we estimate w_{ij} by taking 500 samples of a_j, which are inputs to all the three policies.

Simulation results: In Figure 2(a), we evaluate the performance of the three policies by varying the number of configurations. The migration costs are sampled from $U(0, 1.5)$ and λ_j^{-1} are sampled from $U(1, 2)$ for all j. We observe that for the all three strategies, the time-average cost decreases as the number of configurations increase. This is because the uncertainty to the attacker increases with n. Moreover, the cost of VI decreases much faster than the two baselines, which indicates the weakness of the simple heuristics for large n.

Figure 2(b) compares the average costs of the three policies when the mean attack time λ_j^{-1} is sampled from $U(\nu - 0.5, \nu + 0.5)$ for each j where ν increases from 0.5 to 2.5.



(a) (b) (c)

Figure 2: Simulation Results.

The number of configurations is fixed to 10 and the migration costs are sampled from $U(0.5, 1)$. It is expected that the costs of all the strategies decrease as attack time increases. Again, our algorithm performs much better than the two baselines. Further, the gap increases for large ν. This is because for large attack time, the migration cost becomes the dominant factor, which is not properly taken into account in the two baselines.

In Figure 2(c), we compare the three policies under different variances of the migration cost distribution. In this case, $n = 10$ and λ_j^{-1} is sampled from $U(0.5, 1.5)$ for each j. The migration costs are sampled from $U[\underline{m}, \overline{m}]$ where the mean migration cost $(\underline{m} + \overline{m})/2$ is fixed to 1.5 and we vary $(\underline{m} - \overline{m})/2$. We observe that the performance of PS is close to VI for small variances while the gap becomes bigger for large variances. This is because when each node has a similar migration cost, the loss due to attacks becomes the dominant part in the total cost, which is considered in both PS and VI. On the other hand, when the variance becomes large, our algorithm is able to better handle the heterogeneity of configurations by jointly optimizing P and τ and by considering a different τ_i for each i.

VI. Conclusion

In this paper, we propose a Stackelberg game model for moving target defense (MTD) that jointly considers the spatial and temporal decisions in MTD. In contrast to the $i.i.d.$ strategies considered in most previous works, our model considers the more general Markovian strategies and further incorporates state-dependent attack times. By formulating the defender's problem as a semi-Markovian decision process, we derive a nearly optimal defense strategy that can be efficiently implemented by utilizing the structure of the MTD game.

Acknowledgment

This work has been funded by NSF grant CNS-1816495.

References

[1] "Advanced persistent threat," http://en.wikipedia.org/wiki/Advanced_persistent_threat.

[2] K. D. Bowers, M. E. V. Dijk, A. Juels, A. M. Oprea, R. L. Rivest, and N. Triandopoulos, "Graph-based approach to deterring persistent security threats," US Patent 8813234, 2014.

[3] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "Openflow random host mutation: transparent moving target defense using software defined networking," in *Proc. of HotSDN*, 2012, pp. 127–132.

[4] ——, "Spatio-temporal Address Mutation for Proactive Cyber Agility against Sophisticated Attackers," in *ACM Workshop on Moving Target Defense*, 2014.

[5] B. Salamat, T. Jackson, G. Wagner, C. Wimmer, and M. Franz, "Runtime defense against code injection attacks using replicated execution," *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 4, pp. 588–601, 2011.

[6] L. Szekeres, M. Payer, T. Wei, and D. Song, "SoK: Eternal War in Memory," in *IEEE Symposium on Security and Privacy*, 2013.

[7] C. L. Goues, T. Nguyen, S. Forrest, and W. Weimer, "GenProg: A Generic Method for Automatic Software Repair," *IEEE Transactions on Software Engineering*, vol. 38, no. 1, pp. 54–72, 2012.

[8] A. Nguyen-Tuong, D. Evans, J. C. Knight, B. Cox, and J. W. Davidson, "Security through redundant data diversity," in *Proc. of IEEE DSN*, 2008.

[9] Q. Zhu and T. Başar, "Game-theoretic approach to feedback-driven multi-stage moving target defense," in *Proc. of GameSec*, 2013.

[10] S. Sengupta, S. G. Vadlamudi, S. Kambhampati, A. Doupé, Z. Zhao, M. Taguinod, and G.-J. Ahn, "A game theoretic approach to strategy generation for moving target defense in web applications," in *Proc. of AAMAS*, 2017, pp. 178–186.

[11] X. Feng, Z. Zheng, P. Mohapatra, and D. Cansever, "A Stackelberg Game and Markov Modeling of Moving Target Defense," in *Proc. of GameSec*, 2017.

[12] A. Schlenker, O. Thakoor, H. Xu, F. Fang, M. Tambe, L. Tran-Thanh, P. Vayanos, and Y. Vorobeychik, "Deceiving Cyber Adversaries: A Game Theoretic Approach," in *Proc. of AAMAS 2018*, 2018.

[13] M. Tambe, *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press, 2011.

[14] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience, 1994.

[15] J. Pawlick, E. Colbert, and Q. Zhu, "A Game-Theoretic Taxonomy and Survey of Defensive Deception for Cybersecurity and Privacy," arXiv preprint arXiv:1712.05441, 2017.

[16] R. Zhuang, S. A. DeLoach, and X. Ou, "A model for analyzing the effect of moving target defenses on enterprise networks," in *Proc. of CISR*, 2014.

[17] H. Maleki, S. Valizadeh, W. Koch, A. Bestavros, and M. van Dijk, "Markov modeling of moving target defense games," in *ACM Workshop on Moving Target Defense*, 2016, pp. 81–92.

[18] M. van Dijk, A. Juels, A. Oprea, and R. L. Rivest, "FlipIt: The Game of "Stealthy Takeover"," *Journal of Cryptology*, vol. 26, no. 4, pp. 655–713, 2013.

[19] A. Laszka, B. Johnson, and J. Grossklags, "Mitigating Covert Compromises: A Game-Theoretic Model of Targeted and Non-Targeted Covert Attacks," in *Proc. of WINE*, 2013.

[20] M. Zhang, Z. Zheng, and N. B. Shroff, "A Game Theoretic Model for Defending Against Stealthy Attacks with Limited Resources," in *Proc. of GameSec*, 2015.

[21] Z. Zheng, N. B. Shroff, and P. Mohapatra, "When to Reset Your Keys: Optimal Timing of Security Updates via Learning," in *Proc. of AAAI*, 2017.

[22] P. J. Schweitzer, "Perturbation Theory and Finite Markov Chains," *Journal of Applied Probability*, vol. 5, no. 2, pp. 401–413, 1968.

[23] R. Cavazos-Cadena, "Value iteration and approximately optimal stationary policies in finite-state average Markov decision chain," *Mathematical Methods of Operations Research*, vol. 56, no. 11, pp. 181–196, 2002.

[24] H. Li and Z. Zheng, "Optimal Timing of Moving Target Defense: A Stackelberg Game Model," Technical Report, available online at https://arxiv.org/abs/1905.13293.