Spatial-Temporal Moving Target Defense: A Markov Stackelberg Game Model

Henger Li, Wen Shen, Zizhan Zheng
Department of Computer Science, Tulane University
{hli30,wshen9,zzheng3}@tulane.edu

ABSTRACT

Moving target defense has emerged as a critical paradigm of protecting a vulnerable system against persistent and stealthy attacks. To protect a system, a defender proactively changes the system configurations to limit the exposure of security vulnerabilities to potential attackers. In doing so, the defender creates asymmetric uncertainty and complexity for the attackers, making it much harder for them to compromise the system. In practice, the defender incurs a switching cost for each migration of the system configurations. The switching cost usually depends on both the current configuration and the following configuration. Besides, different system configurations typically require a different amount of time for an attacker to exploit and attack. Therefore, a defender must simultaneously decide both the optimal sequence of system configurations and the optimal timing for switching. In this paper, we propose a Markov Stackelberg Game framework to precisely characterize the defender's spatial and temporal decision-making in the face of advanced attackers. We introduce a value iteration algorithm that computes the defender's optimal moving target defense strategies. Empirical evaluation on real-world problems demonstrates the advantages of the Markov Stackelberg game model for spatial-temporal moving target defense.

KEYWORDS

Moving target defense; Stackelberg game; Markov decision process

ACM Reference Format:
Henger Li, Wen Shen, Zizhan Zheng. 2020. Spatial-Temporal Moving Target Defense: A Markov Stackelberg Game Model . In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), Auckland, New Zealand, May 9–13, 2020,* IFAAMAS, 9 pages.

1 INTRODUCTION

Moving target defense (MTD) has established itself as a powerful framework to counter persistent and stealthy threats that are frequently observed in Web applications [30, 32], cloud-based services [10, 22], database systems [39], and operating systems [7]. The core idea of MTD is that a defender proactively switches configurations (a.k.a., attack surfaces) of a vulnerable system to increase the uncertainty and complexity for potential attackers and limits the resources (e.g., window of vulnerabilities) available to them [8]. In contrast to MTD techniques, traditional passive defenses typically use analysis tools to identify vulnerabilities and detect attacks.

A major factor for a defender to adopt MTD techniques rather than passive defensive applications is that adaptive and sophisticated attackers often have asymmetric advantages of resources (e.g., time, prior knowledge of the vulnerabilities) over the defender due to the static nature of system configurations [19, 29]. For instance, clever attackers are likely to exploit the system over time and subsequently identify the optimal targets to compromise without being detected by the static defensive applications [38]. In such case, a well-crafted MTD technique is more effective because it changes the static nature of the system and increases the attackers' complexity and cost of mounting successful attacks [5]. In this way, it reduces or even eliminates the attackers' asymmetric advantages of the resources.

Despite the promising prospects of MTD techniques, it is challenging to implement optimal MTD strategies. Two factors contribute to the difficult situation. On one hand, the defender must make the switching strategies sufficiently unpredictable because otherwise the attacker can thwart the defense. On the other hand, the defender should not conduct too frequent system migrations because each migration incurs a switching cost that depends on both the current configuration and the following configuration. Thus, the defender must make a careful tradeoff between the effectiveness and the cost efficiency of the MTD techniques. The tradeoff requires the defender to simultaneously decide both the optimal sequence of system configurations (i.e., the next configuration to be switched to) and the optimal timing for switching. To this end, an MTD model must precisely characterize the defender's spatial-temporal decision-making.

A natural approach to model the strategic interactions between the defender and the attacker is to use game-theoretic models such as the zero-sum dynamic game [36] and the Stackelberg Security game (SSG) model [25, 27, 31]. The dynamic game model is general enough to capture different transition methods of the system states and various information structures but optimal solutions to the game are often difficult to compute. This model also assumes that the switching cost from one configuration to another is fixed [36]. The SSG model belongs to an important game-theoretic model called Stackelberg games [33] that has broad applications in many fields, including spectrum allocation [35], smart grid [34] and security [28]. In the SSG model, the defender commits to a mixed strategy that is independent of the state transitions of the system [27, 31]. The attacker adopts a best-response strategy to the defender's mixed strategy. Whenever there is a tie, the attacker always breaks the tie in favor of the defender [18]. This solution concept is called the *strong Stacklberg equilibrium* [3]. While optimal solutions to an SSG can be obtained efficiently, the SSG models neglect the fact that both the defender's and the attacker's strategy can be contingent on the system configuration state. To address

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 9–13, 2020, Auckland, New Zealand.

this problem, Feng et al. incorporate both Markov decision process and Stackelberg game into the modeling of the MTD game [6].

Many MTD models [4, 9, 25, 27] do not explicitly introduce the concept of the defending period, although they usually assume that the defender chooses to execute a migration after a constant time period [26]. A primary reason is that when both time and the system's state influence the defender's decision making, optimal solutions to the defender's MTD problem are non-trivial [15, 26]. Recently, Li and Zheng has proposed to incorporate timing into the defender's decision making processes [14]. Their work assumes that there is a positive transition probability between any two configurations (so that the corresponding Markov decision process is unichain), which may lead to sub-optimal MTD strategies. Further, their model assumes that all the attackers have the same type, which might not be true in reality. To solve the defender's optimal MTD problem in more general settings, an MTD model that precisely models the strategic interactions between the defender and the attacker is in urgent need.

Our contributions. In this paper, we propose a general framework called the *Markov Stackelberg Game* (MSG) model for spatial-temporal moving target defense. The MSG model enables the defender to implement optimal defense strategy that is contingent on both the source state and the destination state of the system. It also allows the defender to simultaneously decide which state the system should be migrated to and when it should be migrated. In the MSG model, we formulate the defender's optimization problem as an average-cost semi-Markov decision process (SMDP) [23] problem. We present a value iteration algorithm that can solve the average-cost SMDP problem efficiently after transforming the original average-cost SMDP problem into a discrete time Markov decision process (DTMDP) problem. We empirically evaluate our algorithm using real-world data obtained from the National Vulnerability Database (NVD) [20]. Experimental results demonstrate the advantages of using MSG over the state-of-the-art approaches in MTD.

2 MODEL

In this section, we describe the Markov Stackelberg Game (MSG) model for moving target defense.

2.1 System Configuration

In moving target defense, a defender proactively shifts variables of a computing system to increase uncertainty for the attacker to mount successful attacks. The variables of the system are often called adaptation aspects [37, 38]. Typical adaptation aspects include IP address, port number, network protocol, operating system, programming language, machine replacement, and memory to cache mapping [37]. A computing system usually has multiple adaptation aspects. Let D denote the number of adaptation aspects for the computing system and Φ_i the set of sub configurations in the i-th adaptation aspect. If the defender selects the sub configuration $\phi_i \in \Phi_i$ for the i-th adaptation aspect, then the configuration state of the system is denoted as $s = (\phi_1, \phi_2, \ldots, \phi_D)$. Here, s is a generic element of the set of system configurations $S = \Phi_1 \times \Phi_2 \times \cdots \times \Phi_D$. Let $n = |S|$ denote the number of system configurations in S. We use *configuration* and *state* interchangeably throughout the paper.

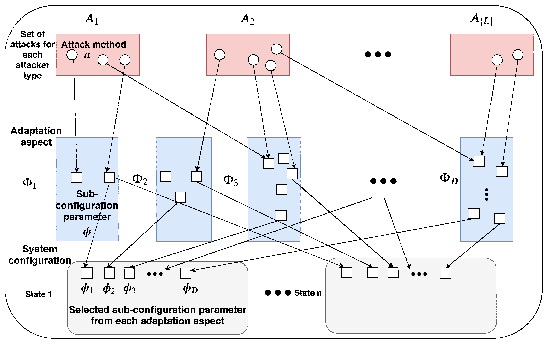

Figure 1: An illustration of the relationship between attacker types, attack methods, adaptation aspects, sub-configuration parameters, and system configurations.

2.2 The Attacker

2.2.1 Attacker Type. A successful attack requires a competent attacker that has the expertise to exploit the vulnerabilities. For instance, to gain control of a remote computer, the attacker needs the capability to obtain the control of at least one of the following resources: the IP address of the computer, the port number of a specific application running on the machine, the operating system type, the vulnerabilities of the application, or the root privilege of the computer [37]. The attacker's ability profile is called the *attacker type*. There are different attacker types. Let L denote the set of attacker types. Each attacker type $l \in L$ is capable of mounting a set of attacks A_l. The attacker type space A_l is a nonempty set of the attack methods that each targets one vulnerability in one adaption aspect, which may affect multiple sub configurations in that aspect. Multiple attacks may target the same vulnerability but may have different benefit (loss) to the attacker (defender). Whether an attack method belongs to an attacker type space or not depends on the application scenarios. Figure 1 illustrates the relationship between attacker types, attack methods, adaptation aspects, sub-configuration parameters and system configurations.

2.2.2 Attacking Time. If an attacker with the attacker type l chooses an attack method $a \in A_l$ to attack the system in state j, then the time needed for him to compromise the system is a random variable $\xi_{a,j}$ that is drawn from a given distribution $\Xi_{a,j}$. If a is not targeting a vulnerability of any sub configuration in state j, $\xi_{a,j} = +\infty$. The attacker only gains benefits when he has successfully compromised the system. A system is considered to be compromised as long as the attacker compromises one of the sub configurations. In this work, L, $\{A_l\}$, and $\{\Xi_{a,j}\}$ are assumed to be common knowledge.

2.3 The Defender

2.3.1 Migration Cost. During the migration process, the defender updates the system to retake or maintain the control of the system and pays some updating cost. The defender then selects and shifts the system from the current state $i \in S$ to the next valid system state $j \in S$ with a cost m_{ij}. We can consider that the migration is implemented instantaneously. This is without loss of

generality because one can always assume that during the migration the system is still at state i. If the defender decides to stay at the current state i, then the cost is m_{ii}. Since there is a cost for updating the system, it requires that $m_{ii} > 0$ for all $i \in S$. Let M be the matrix of the migration cost between any states $i, j \in S$, we have $M = [m_{ij}]_{n\times n}$. It is crucial to note that the migration cost m_{ij} may depend on both the *source* state i and the *destination* state j.

2.3.2 Defending Period. The defender not only needs to determine which state to use but also should decide when to move. If the defender stays in a state sufficiently long, the attacker is likely to compromise the system even if the defender shifts eventually. Thus, the defender needs to pick the defending period judiciously. Let t_k denote the time instance that the k-th migration happens. The k-th defending period is calculated by $t_k - t_{k-1}$. In practice, the system needs some time to prepare migration or updating. Thus, it is natural to require that the defending period is lower bounded. On the other hand, the defending period cannot be arbitrarily large because otherwise the system will be compromised eventually and stay compromised afterwards. Therefore, we assume that $\tau \in [\underline{\tau}, \overline{\tau}]$ where $0 < \underline{\tau} < \overline{\tau}$.

2.3.3 Defender's Strategy. In our model, the defender adopts a stationary strategy $\mathbf{u}$ that simultaneously decides where to move and when to move. Let p_{ij} denote the probability that the defender moves from state i to state j, then the transition probabilities between any two states in S can be represented by a transition matrix $P = [p_{ij}]_{n\times n}$, where $\sum_{j\in S} p_{ij} = 1$ for any $i \in S$. When the system state is i, the defender uses a mixed strategy $\mathbf{p}_i = (p_{ij})_{j\in S}$ to determine the next state to switch to. It is crucial to note that the transition probabilities depend on both the source and destination configurations.

Moreover, instead of a fixed defending period as in most MTD models, we consider a state-dependent defending period in our model. In particular, let τ_i denote the length of the $(k+1)$-th defending period when the system is in state i in the k-th defending period. The defending period matrix is thus $\boldsymbol{\tau} = [\tau_1, \tau_2, \ldots, \tau_n]$. In our model, we assume that the $(k+1)$-th defending period only depends on the system state in the k-th period due to two considerations. First, the transition probabilities have already captured the dependencies on destination states. Second, it allows the defender's problem to be modeled as a semi-Markov decision process (defined in Section 2.4.3). The defender's strategy is then denoted by $\mathbf{u} = (P, \boldsymbol{\tau}) = [u(i)]_{i\in S}$ where $u(i) = (\mathbf{p}_i, \tau_i)$ is the defender's action when the system is in state i. See Figure 2 for an illustration of the system model.

2.4 Markov Stackelberg Game

In our paper, we model moving target defense as a Markov Stackelberg game (MSG) where the defender is the *leader* and the attacker is the *follower*. At the beginning the defender commits to a stationary strategy that is announced to the attacker. The attacker's goal is to maximize his expected reward by constantly attacking the system's vulnerabilities. The defender's objective is to minimize the total loss due to attacks plus the total migration cost.

The MSG model is an extension of the Stackelberg Security Game (SSG) model that has been widely adopted in security domains [28].

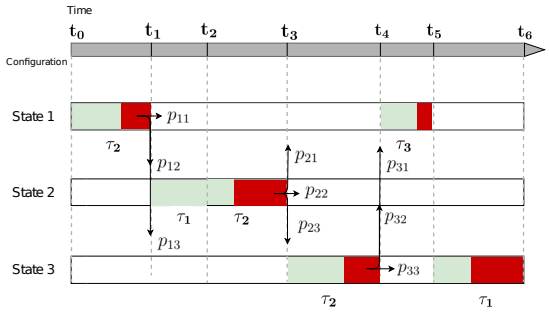

Figure 2: An illustration of the Markov Stackelberg game model. A light green block represents the time period when the system is protected while a dark red block denotes the time period when the system is compromised; p_{ij} is the probability that the defender moves from state i to j, and τ_i is the length of the current defending period when the previous configuration is i. The first defending period (between t_0 and t_1) depends on the initial configuration (State 2 in this case).

A key advantage of using the Stackelberg game model is that it enables the defender to act first to commit to a mixed strategy that the attacker observes and then selects his best response accordingly [12]. This advantage allows the defender to implement defense strategies prior to potential attacks [14]. In MTD, the defender proactively switches the system between different configurations to increase the attacker's uncertainty. It is thus natural to model the defender as the leader and the attacker as the follower.

2.4.1 Information Structure. In our model, we consider a stealthy and persistent attacker that learns the defender's stationary strategy $\mathbf{u}$. We assume that the defender announces her stationary strategy to the attacker before the game starts. This is without loss of generality because the attacker will learn the system states and the defender's strategy eventually due to the stealthy and persistent nature of the attacks. We further assume that the attacker always learns the system state at the end of a defending period no matter the attack is successful or not in that period. Hence, at the beginning of each stage, the only uncertainty to the attacker is the current state as he has already known the previous state and the length of the current defending period. This is a worse-case scenario from the defender's perspective.

In many security domains, it is often difficult for the defender to obtain real-time feedback about whether the system is compromised or not because the attacker is likely to be stealthy. The defender, however, may have some prior knowledge about the attacker type distribution [21, 27]. Thus, we assume that the defender has a prior belief $\pi = \{\pi_l\}$ on the probability distribution of the attacker type l.

2.4.2 Attacker's Optimization Problem. Since the defender adopts a stationary strategy that is known to the attacker, the attacker can maximize its long-term payoff by always following a best response in each stage. Hence, we consider a myopic attacker that always adopts a best response to the defender's strategy in each defending period according to its knowledge on the previous system state.

Consider the beginning of the k-th defending period from the attacker's perspective. Assume that the system was in state i (known to the attacker) in the $(k-1)$-th period and has moved to state j (unknown to the attacker) in the k-th period. For an attack method a to be successful in the k-th period, the time $\xi_{a,j}$ required for the attacker when using attack method a targeting state j should be less than the length of the defending period τ_i. Let $R^l_{a,j}$ denote the attacker's benefit per unit of time when the system is compromised, which is jointly determined by the attacker's type l, its chosen attack method a, and the state j. The reward that the attacker receives in the k-th period is then $(\tau_i - \xi_{a,j})^+ R^l_{a,j}$ where $(x)^+ \triangleq \max(0, x)$. Since the attacker only knows i but not j when it chooses the attack method at the beginning of the k-th defending period, the optimization problem for the attacker with type l is to maximize his expected reward by choosing an attack method a from his attack space A_l: $\max_{a\in A_l} \sum_{j\in S} p_{ij}\mathbb{E}[(\tau_i - \xi_{a,j})^+]R^l_{a,j}$, where τ_i and $\{p_{ij}\}$ are the defending period length and the transition probabilities given by the defender under state i, respectively.

2.4.3 Defender's Optimization Problem. We consider a defender that constantly migrates among the system configurations in order to minimize the total loss due to attacks plus the total migration cost. We use an average cost semi-Markov decision process (SMDP) to model the defender's optimization problem. The SMDP model considers a defender that aims to minimize her long-term defense cost in an infinite time horizon using spatial-temporal decision making.

Consider the end of the $(k-1)$-th defending period from the defender's perspective. Again assume that the system is in state i in the $(k-1)$-th period. Given the defender's action $u(i) = (\mathbf{p}_i, \tau_i)$, her expected cost in the k-th defending period is:

$$c(i, u(i)) = \sum_{l\in L} \pi_l \left(\sum_{j\in S} p_{ij}\mathbb{E}[(\tau_i - \xi_{a^l,j})^+]C^l_{a^l,j} \right) + \sum_{j\in S} p_{ij} m_{ij}$$

$$s.t. \quad a^l = \arg\max_{a\in A_l} \left(\sum_{j\in S} p_{ij}\mathbb{E}[(\tau_i - \xi_{a,j})^+]R^l_{a,j} \right), \quad \forall l \in L \quad (1)$$

where $C^l_{a,j}$ denotes the unit time loss for the defender under state j due to an attack a launched by a type l attacker. The first part in the objective function is the expected attacking loss and the second part is the expected migration cost.

Let s_0 be the initial system configuration before the game starts, which is arbitrarily chosen from the state space S and is known to the attacker. The game starts at $t_0 = 0$ when the defender applies the first migration. Let s_k denote the system state in the k-th defending period for $k = 1, 2, \ldots$. The defender adopts a stationary policy u where $u(s_k) = (\mathbf{p}_{s_k}, \tau_{s_k})$ for each state s_k, which generates an expected cost $c(s_k, u(s_k))$ that includes potential loss from compromises and migrations. Given the initial state s_0, the defender's long-term average cost is defined as:

$$z(s_0, u(s_0)) = \limsup_{N\to\infty} \frac{\sum_{k=0}^{N-1} c(s_k, u(s_k))}{\sum_{k=0}^{N-1} \tau_{s_k}} \quad (2)$$

The goal of the defender is to commit to a stationary policy $\mathbf{u}^* = [u^*(i)]_n$ that minimizes the time-average cost for any initial state. For each $u(i) = (\mathbf{p}_i, \tau_i)$, we have $p_{ij} \in [0, 1]$ for all j, $\sum_j p_{ij} = 1$, and $\tau_i \in [\underline{\tau}, \overline{\tau}]$. Thus, the action space $U(i)$ for every $u(i)$ is $[0, 1]^n \times [\underline{\tau}, \overline{\tau}]$, which is a continuous space. We assume that $c(i, u(i))$ is continuous over $U(i)$. The defender's optimization problem corresponds to finding a strong Stackelberg equilibrium [3] where the defender commits to an optimal strategy assuming that the attacker will choose the best response to the defender's strategy and break ties in favor of the defender. This is a common assumption in Stackelberg security game literature.

2.4.4 Challenges of Computing Optimal MTD Strategies. The are two main challenges for the defender to compute the optimal strategies. First, when an arbitrary transition matrix P is allowed, the Markov chain associated with the given P may have a complicated chain structure. The optimal solution may not exist and standard methods such as Value Iteration (VI) and Policy Iteration (PI) [23] may never converge when applied to an average-cost SMDP with a continuous action space. Second, a bilevel optimization problem needs to be solved in each iteration of VI or PI, which is challenging due to the infinite action space and the coupling of spatial and temporal decisions.

2.5 MSG versus SSG

Our Markov Stackelberg game model extends the classic Stackelberg Security Game (SSG) [21] in important ways. In the classic SSG model, there is a set of targets and a defender has a limited amount of resources to protect them. The defender serves as the leader and commits to a mixed strategy. The attacker observes the defender's strategy (but not her action) and then responds accordingly. Thus, the SSG model is essentially a one-shot game. The SSG model has been extended to Bayesian Stackelberg Game (BSG) model to capture multiple attack types where the defender knows the distribution of attack types a prior as we assumed. In a recent work [27], the BSG model has been used to model moving target defense where only the spatial decision is considered and the defender commits to a vector $[p_j]_n$ where p_j is the probability of moving to state j in the next stage, which is independent of the current state.

We note that the BSG model in [27] is a special case of our MSG model. Specifically, let $\tau_i = 1$ for all $i \in S$, $\xi_{a^l,j} = 0$ for all $a^l \in A^l$, $l \in L$, $j \in S$, and $p_{ij} = p_j$ for all $i, j \in S$. Then the SMDP becomes an MDP with state independent transition probabilities. Since each row of the transition matrix is the same, the stationary distribution of the corresponding Markov chain is just $[p_j]_n$. Therefore, the average-cost SMDP reduces to the following one-stage optimization problem to the defender:

$$\min_{\mathbf{p}} \sum_l \pi_l \Big(\sum_j p_j C^l_{a^l,j} \Big) + \sum_{i,j} p_i p_j m_{ij}$$

$$s.t. \quad a^l = \arg\max_{a\in A_l} \Big(\sum_j p_j R^l_{a,j} \Big), \quad \forall l \in L \quad (3)$$

This is exactly the BSG model for MTD in [27]. In the BSG variant, the transition probabilities depend on the destination state only. This simplified MTD strategy is optimal only if the migration cost depends on the destination state only but not the source state.

Our MSG model enables the defender to handle the complex scenarios when the migration cost is both source and destination

dependent. It also takes the defending period into account in computing the optimal defense strategy. This consideration is useful because a stealthy and persistent attacker will compromise the system eventually if the system stays in a state longer than the corresponding attacking time.

3 OPTIMAL MOVING TARGET DEFENSE

3.1 Assumptions

Assumption 1. *The transition probability matrix P can be arbitrarily chosen by the defender.*

Assumption 2. *For any $i \in S$, the defender's cost per unit time $c(i, u(i))/\tau_i$ is continuous and bounded over $U(i)$.*

Both assumptions are reasonable and can be easily satisfied. Assumption 1 implies an important structure property of the SMDP as formally defined below.

Definition 3.1 (Communicating MDP [23]). For every pair of states i and j in S, there exists a deterministic stationary policy $\mathbf{u}$ under which j is accessible from i, that is, $\Pr(s_k = j | s_0 = i, \mathbf{u}) > 0$ for some $k \geq 1$.

It is easy to check that the SMDP in our problem is communicating under Assumption 1. It is known that for a communicating MDP, the optimal average cost is a constant, independent of the initial state [23]. This property significantly simplifies the algorithm design and analysis as we discuss below. Assumption 2 is used in establishing the convergence of the value iteration algorithm under the continuous action space (see Section 3.3.3).

3.2 Data Transformation

Solving the defender's optimization problem requires the algorithm to simultaneously determine the optimal transition probabilities and the optimal defending periods. The average-cost SMDP problem with continuous action space is known to be difficult to solve [11]. Fortunately, one can apply the data transformation method introduced by Schweitzer [24] to transform the average-cost SMDP problem into a discrete-time average Markov decision process (DTMDP) problem. The DTMDP has a simpler structure than the SMDP with the same state space S and action space $U(i)$ for any $i \in S$. The defender's per-stage cost $c(i, u(i))$ is converted to

$$\tilde{c}(i, u(i)) = \frac{c(i, u(i))}{\tau_i} \tag{4}$$

Further, the transition probability from state i to state j for the DTMDP is

$$\tilde{p}_{ij}(u(i)) = \gamma \frac{p_{ij} - \delta_{ij}}{\tau_i} + \delta_{ij} \tag{5}$$

where δ_{ij} denotes the Kronecker delta (i.e., $\delta_{ii} = 1$ and $\delta_{ij} = 0$ for all $j \neq i$) and γ is a parameter that satisfies $0 < \gamma < \underline{\tau} \leq \frac{\tau_i}{1-p_{ii}}$, where $\underline{\tau}$ is the lower bound of the defending period length. Let $\tilde{P}(\mathbf{u}) = [\tilde{p}_{ij}(u(i))]_{n \times n}$ denote the transition probability matrix of the DTMDP and $\tilde{\mathbf{c}}(\mathbf{u}) = [\tilde{c}(i, u(i))]_n$ the defender's per-stage cost across all the states. If the system starts from the initial state $s_0 \in S$, then the long-term average cost becomes $\tilde{z}(s_0, u(s_0)) = \limsup_{N \to \infty} \frac{1}{N} \sum_{k=0}^{N-1} \tilde{c}(s_k, u(s_k))$.

The above data transformation has some nice properties as summarized below.

Theorem 3.2 (Theorems 5.2 and 5.3 of [11]). *Suppose an SMDP is transformed into a DTMDP using the above method. We have*

1. *If SMDP is communicating, then DTMDP is also communicating.*
2. *If SMDP is communicating, then a stationary optimal policy for DTMDP is also optimal for SMDP.*

Theorem 3.2 indicates that the transformed DTMDP also has a contant optimal cost and further, to find a stationary optimal policy for the SMDP in our problem, it suffices to find a stationary optimal policy for the transformed DTMDP.

3.3 Value Iteration Algorithm

3.3.1 Additional Notations. Let V be any vector in $\mathbb{R}^n$. We define the mapping $F : \mathbb{R}^n \to \mathbb{R}^n$ as: $F(V) = \min_{\mathbf{u}} [\tilde{\mathbf{c}}(\mathbf{u}) + \tilde{P}(\mathbf{u})V]$, where the minimization is applied to each state i separately. For any vector $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, let $L(\mathbf{x}) = \min_{i=1,\ldots,n} x_i$ and $H(\mathbf{x}) = \max_{i=1,\ldots,n} x_i$. Let $\|\cdot\|$ denotes the span seminorm defined as follows: $\|\mathbf{x}\| = H(\mathbf{x}) - L(\mathbf{x})$. It is easy to check that $\|\cdot\|$ satisfies the triangle inequality, that is, $\|\mathbf{x} - \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$ for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$. Further, $\|\mathbf{x} - \mathbf{y}\| = 0$ if and only if there exists a scalar λ such that $\mathbf{x} - \mathbf{y} = \lambda \mathbf{e}$ where $\mathbf{e}$ is the n-dimensional vector of all ones. Thus, there is a vector V such that $\|F(V) - V\| = 0$ (V is called a fixed point of $F(\cdot)$) if and only if there is a scalar λ such that the following optimality equation is satisfied:

$$\lambda \mathbf{e} + V = \min_{\mathbf{u}} [\tilde{\mathbf{c}}(\mathbf{u}) + \tilde{P}(\mathbf{u})V] \tag{6}$$

An important result in MDP theory [2, 23] is that the stationary policy $\mathbf{u}$ that attains the minimum in the optimality equation (6) is optimal and λ gives the optimal long-term average cost.

3.3.2 The VI Algorithm. The VI algorithm (see Algorithm 1) maintains a vector $V^t \in \mathbb{R}^n$. The algorithm starts with an arbitrary V^0 (line 1) and a carefully chosen sequence $\{\kappa_t\}$ (line 2) that ensures every limit point of $\{V^t\}$ is a fixed point of $F(V)$ (See Section 3.3.3). In each iteration, V^t is updated by solving a policy improvement step (line 5).

In each policy improvement step (see Algorithm 2), instead of finding the optimal $\mathbf{p}_i$ and τ_i together for each state i, which is a challenging problem, we discretize $[\underline{\tau}, \overline{\tau}]$ and search for τ_i with a step size δ (line 3). This approximation is reasonable since in practice the unit time cannot be infinitely small. The smaller δ is, the closer τ^* (line 9 in Algorithm 1) is to the optimal one. Also note that the optimization problem in lines 4 is actually a bilevel problem, which can be modeled as a Mixed Integer Quadratic Program (MIQP) and will be discussed in detail in Section 3.4.

Under Assumptions 1 and 2, Algorithm 1 stops in a finite number of iterations (lines 3-8) and is able to find a near-optimal policy (formally proved in Section 3.3.3). In practice, a near-optimal solution is sufficient because it can be expensive or even unrealistic to obtain the exact minimum average cost in a large-scale MDP. The algorithm itself, however, can attain the optimal solution if the number of iterations goes to infinity (and δ approaches 0).

Remark: Algorithm 1 can be slow in practice due to the large number of iterations needed to converge and the complexity of solving multiple MIQP problems in each iteration. To obtain a more efficient

Algorithm 1 Value Iteration algorithm for the MTD game

Input: $S, n, \epsilon > 0, \underline{\tau}, \overline{\tau}, M, C, R, \pi, L, \{A_l\}, \delta > 0$.
Output: P^*, τ^*
1: $V^0 \in \mathbb{R}^n$;
2: $\kappa_0 = \frac{1}{2}$, $\kappa_t = \frac{t}{t+1}$ for $t = 1, 2, \ldots$.
3: **repeat**
4: $t = t + 1$;
5: $V^t = \textbf{PImp}(S, V^{t-1}, \underline{\tau}, \overline{\tau}, M, C, R, \pi, L, \{A_l\}, \delta, \kappa_{t-1})$
6: $\overline{V} = \max_{i \in S} |V^t(i) - V^{t-1}(i)|$;
7: $\underline{V} = \min_{i \in S} |V^t(i) - V^{t-1}(i)|$;
8: **until** $\overline{V} - \underline{V} < \epsilon$
9: $P^*, \tau^* = \arg \textbf{PImp}(S, V^{t-1}, \underline{\tau}, \overline{\tau}, M, C, R, \pi, L, \{A_l\}, \delta)$

Algorithm 2 Policy Improvement (**PImp**)

Input: $S, V_0, \underline{\tau}, \overline{\tau}, M, C, R, \pi, L, \{A_l\}, \delta, \kappa$
Output: V
1: **for** $i \in S$ **do**
2: $v = +\infty$;
3: **for** $\tau = \underline{\tau}; \tau \le \overline{\tau}; \tau = \tau + \delta$ **do**
4: $V(i) = \min_{\mathbf{p}_i} [\tilde{c}(i, \mathbf{p}_i, \tau) + \kappa \sum_{j \in S} \tilde{p}_{ij}(\mathbf{p}_i, \tau) V_0(j)]$
5: **if** $V(i) < v$ **then**
6: $v = V(i)$;
7: **end if**
8: **end for**
9: $V(i) = v$;
10: **end for**

solution, we have considered a variant of Algorithm 1 using the relative value iteration (RVI) technique (see our technical report [13]). Although the RVI variant does not improve the theoretic convergence speed of the VI algorithm, it often requires fewer iterations for the same ϵ in practice. We have further made the following observation. If we introduce the assumption that $p_{is} \ge \rho > 0$ for all $i \in S$ (where ρ can be arbitrarily small) for some fixed state s to restrict the Markov chain structure and set $\kappa_t = 1$ for all t, the rate of convergence can be significantly improved both in theory [2] and in practice for both the VI and the RVI algorithms. We conjecture that a near-optimal policy can still be obtained by making ρ small enough.

3.3.3 Theoretical Analysis. Our VI algorithm is adapted from the work due to Bertsekas [2] and Bather [1] that originally addresses the average cost MDP problem with a finite state space and an infinite action space. However, their proofs do not directly apply to our algorithm because they either consider the transition probabilities as the only decision variables [1] or involve the use of randomized controls [2]. In contrast, our strategy includes both the probability transition matrix and the (deterministic) defending periods.

For a given stationary policy $\mathbf{u}$ with transition matrix $\tilde{P}$, let $\tilde{P}^*$ denote the Cesaro limit given by $\tilde{P}^* = \lim_{N \to \infty} \{I + \tilde{P} + \tilde{P}^2 + \cdots + \tilde{P}^{N-1}\}/N$. Then the average cost associated with $\mathbf{u}$ can be represented as $\tilde{P}^* \tilde{c}(\mathbf{u})$[23]. The policy $\mathbf{u}$ is called *ϵ-optimal* if $\tilde{P}^* \tilde{c}(\mathbf{u}) \le \lambda + \epsilon \mathbf{e}$ where λ is the optimal cost vector. In practice, it is often expensive or even unrealistic to compute an exact optimal policy and an ϵ-optimal policy might be good enough. Our main results can be summarized as follows.

Theorem 3.3. *Under Assumptions 1 and 2, we have*

(1) *The DTMDP problem (thus the SMDP problem too) has an optimal stationary policy;*
(2) *The sequence of policies in Algorithm 1 eventually leads to an ϵ-optimal policy.*

Proof Sketch: The first part can be proved using the similar techniques in the proofs of Theorem 2.4 of [1] and Proposition 5.2 of [2]. The main idea is to show that (1) $\{\|V^t\|\}$ is bounded thus the vector sequence $\{V^t\}$ must have a limit point; (2) every limit point of $\{V^t\}$ is a fixed point of $F(\cdot)$, thus leading to an optimal solution. The second part follows from Theorem 6.1 and Corollary 6.2 of [1]. The main idea is to show that (1) if $\|F(V^t) - V^t\| \le \epsilon$, then the corresponding policy is ϵ-optimal; (2) $\lim_{t \to \infty} \|F(V^t) - V^t\| = 0$ (again using the boundedness of $\{\|V^t\|\}$) so that $\|F(V^t) - V^t\| \le \epsilon$ holds eventually. Note that this condition is exactly the stopping condition $\overline{V} - \underline{V} < \epsilon$ in Algorithm 1 (line 8).

The proof of Theorem 3.3 relies on the key property that the vector sequence $\{\|V^t\|\}$ generated by Algorithm 1 is bounded. Due to the coupling of spatial and temporal decisions in our problem, the techniques in [1, 2] cannot be directly applied to prove this fact.

Lemma 3.4. *Let Assumptions 1 and 2 hold and $\{\kappa_t\}$ be a nondecreasing sequence with $\kappa_t \in [0, 1]$ for each t. Consider a sequence $\{V^t\}$ where $V^{t+1} = F(\kappa_t V^t) = \min_{P,\tau} [\tilde{\mathbf{c}}(P, \tau) + \kappa_t \tilde{P}(P, \tau) V^t]$, then $\{\|V^t\|\}$ is bounded.*

Proof Sketch: For a communicating system, for each pair of states i and j, there exists a stationary policy $\mathbf{u}_{ij}$ (with transition matrix $\tilde{P}(\mathbf{u}_{ij})$) such that j is accessible from i. The main idea of the proof is to show that the combined matrix $Q = \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n \tilde{P}(\mathbf{u}_{ij})$ is still a valid transition probability matrix. That is, there exists a policy $\mathbf{u}$ such that $Q = \tilde{P}(\mathbf{u})$. This follows from transformations (4) and (5) and the fact that defender can pick an arbitrary transition matrix. Using this property, the fact that $\{\|V^t\|\}$ is bounded can then be proved by induction. The detailed proof can be found in our online technical report [13].

3.4 Bilevel Optimization Problem

To compute the optimal value in each iteration with Algorithm 2, we need to solve the following optimization problem for a given scalar τ and a vector V^{t-1} (line 4 in Algorithm 2): $V^t(i) = \min_{\mathbf{p}_i} [\tilde{c}(i, \mathbf{p}_i, \tau) + \kappa_{t-1} \sum_{j \in S} \tilde{p}_{ij}(\mathbf{p}_i, \tau) V^{t-1}(j)]$. Substitute $\tilde{c}(i, \mathbf{p}_i, \tau)$ and $\tilde{p}_{ij}(\mathbf{p}_i, \tau)$ by their definitions in Equations (4) and (5), and denote $w_{j,a} \triangleq E[(\tau_i - \xi_{a,j})^+]$ and $\theta_j \triangleq m_{ij} + \gamma \kappa_{t-1} V^{t-1}(j)$ to simplify the notation. Using the similar technique for solving Bayesian Stackelberg games [21], the defender's bilevel optimization problem in (3) can be modeled as the following Mixed Integer Quadratic Program

(MIQP):

$$\min_{\mathbf{p}_i,\mathbf{n},\mathbf{v}} \sum_{j\in S}\sum_{l\in L}\sum_{a\in A_l} \pi_l w_{j,a} C^l_{a,j} p_{ij} n^l_a + \sum_j p_{ij}\theta_j$$

$$s.t. \quad \sum_{j\in S} p_{ij} = 1, \quad \sum_{a\in A_l} n^l_a = 1, \quad \forall l \in L$$

$$0 \le v^l - \sum_j p_{ij} w_{j,a} R^l_{a,j} \le (1 - n^l_a)B, \quad \forall a \in A_l, l \in L \qquad (7)$$

$$p_{ij} \in [0,1], \quad n^l_a = \{0,1\}, \quad v^l \in \mathbb{R}, \quad \forall j \in S, a \in A_l, l \in L$$

where the binary variable $n^l_a = 1$ if $a \in A_l$ is the best action for the type l attacker and $n^l_a = 0$ otherwise. This is ensured by constraint (7) where v^l is an upper bound on the attacker's reward and B is a large positive number.

4 EMPIRICAL EVALUATION

We conducted numerical simulations using the data from the National Vulnerability Database (NVD) [20] to demonstrate the advantages of using MSG for spatial-temporal MTD. In particular, we derived the key attack/defense parameters from the Common Vulnerabilities Exposure (CVE) scores [16] in NVD, which has been widely used to describe the weakness of a system with respect to certain risk levels. We used data samples in NVD with CVE scores ranging from January 2013 to August 2016. As in [27], the Base Scores (BS) and Impact Scores (IS) were used to represent the attacker's reward (per unit time) and the defender's cost (per unit time) respectively. Further, we used the Exploitability Scores (ES) to estimate the distribution of attack time.

We conducted two groups of experiments[1]: the spatial decision setting and the joint spatial-temporal decision setting. We compared the MSG method with two benchmarks: the Bayesian Stackelberg Game (BSG) model [27] and the Uniform Random Strategy (URS) [27]. We used the Gurobi solver (academic version 8.1.1) for the MIQP problems in BSG and MSG. All the experiments were run on the same 24-core 3.0GHz Linux machine with 128GB RAM.

4.1 Spatial Decision

4.1.1 Benchmarks and Settings: In the spatial decision setting, the defender periodically moves in unit time length and the attacker instantaneously compromises the system when he chooses the correct configuration. We compared the MSG model with the original BSG and URS models in [27]. In BSG, the defender determines the next configuration according to a fixed transition probability vector $[p_j]_n$ that is independent of the current configuration. In URS, the defender selects the next configuration uniform randomly.

For fair comparisons, we followed the same data generation method as used in the work by Sengupta et al. [27]. The system has four configurations and the corresponding switching cost is shown in Figure 3. We added an updating cost of 2 to all the switching cost in [27]. In this experiment, we considered three attacker types: the Script Kiddie that could attack *Python* and *PHP*, the Database Hacker that is able to attack *MySQL* and *postgreSQL* and the Mainstream Hacker that could attack all the techniques. The defender possesses a prior belief of $(0.15, 0.35, 0.5)$ on the three attacker types. The number of attack methods for each of these attacker types is 34, 269 and 48, respectively.

[1]Code is available at https://github.com/HengerLi/SPT-MTD.

	PHP, MySQL	Python, MySQL	PHP, postgreSQL	Python, postgreSQL
PHP, MySQL	2	4	8	12
Python, MySQL	4	2	11	7
PHP, postgreSQL	8	11	2	12
Python, postgreSQL	12	7	12	2

Figure 3: The migration cost in the MTD system. Each row represents a source configuration and each column represents a destination configuration.

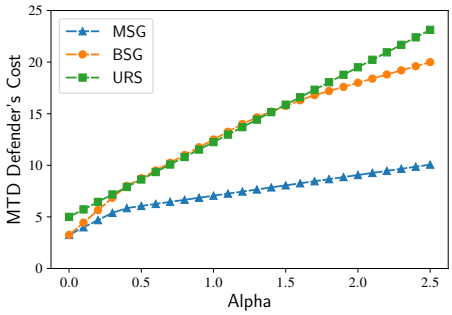

Figure 4: A comparison of the defender's cost in the three policies with spatial decisions only - MSG ($\epsilon = 0.1$), BSG and URS with unit defending period ($\tau_i = 1$ for all $i \in S$) and zero attacking time as the parameter α increases.

For BSG, the defender's optimization problem was directly solved with MIQP as in [27]. For MSG, the bi-level optimization problem was solved with MIQP for each configuration in every iteration of Algorithm 1 (with the convergence parameter $\epsilon = 0.1$). Since the simulated migration cost could not be directly compared with the attacking cost from NVD, we introduced a parameter α to adjust the ratio between the attacking cost and the migration cost. That is, instead of the m_{ij} shown in Figure 3, we used αm_{ij} as the migration cost from state i to state j. As α increases, the migration cost has a larger impact on the defender's decisions.

4.1.2 Results: We varied the value of α from 0 to 2.5 with an increment of 0.1 and compared the defender's cost for the three policies: the MSG, the BSG, and the URS (see Figure 4). All three models were restricted to unit length defending period and zero attacking time. That is $\tau_i = 1$ for all $i \in S$, and $\xi_{a,j} = 0$ for all $a \in A^l, l \in L, j \in S$.

Figure 4 shows that the defender's cost increases for all the three policies as the migration cost grows. However, the magnitude of increase differs in the three policies. In URS, the cost increases linearly due to the uniform random strategy $(0.25, 0.25, 0.25, 0.25)$ used. In both MSG and BSG, the defender's cost grows sub-linearly. However, the defender incurs substantially less cost in MSG. The

reason is that although both MSG and BSG enable the defender to choose the respective optimal strategies, MSG allows the defender to vary her strategy according to different source configurations while in BSG the defender must choose the same strategy for all the source configurations (the detail strategies are shown in our technical report [13]). Further, MSG allows both absorbing states (when the system moves in, it always stays there) and transient states (once the system moves out, it never comes back) leading to more flexible system dynamics, which contributes to the lower defending cost. However, these states cannot be achieved by BSG since the only way to always stay in a configuration under BSG is to assign a probability of 1 to that configuration and the only way to never come back to a configuration is to assign a probability of 0 to it, which, however, removes any uncertainty to the attacker. This indicates that MSG can achieve a better trade-off between the migration cost and the loss from attacks.

4.2 Joint Spatial-Temporal Decision

4.2.1 Benchmarks and Settings: In the joint spatial-temporal decision setting, the defender needs to decide not only the next configuration to move to but also the length of each defending period τ. In our experiments, τ is in the range of $[0.1, 2.6]$ with an increment parameter $\delta = 0.1$. For MSG, the optimal τ_i for each configuration i was obtained together with the spatial decisions using Algorithm 1. We extended the BSG and URS policies by incorporating the attacking times and the defending periods into the objectives of both the defender and the attacker (as we did in our MSG model), where an identical defending period is used for all the configurations since both policies ignore the source configuration in each migration. To have a fair comparison with MSG, we searched for the optimal defending period for BSG and URS respectively.

We assigned a random attacking time for each attack that aims to compromise the system. The random attacking time $\xi_{a,j}$ was drawn from the exponential distribution $Exp(ES_a)$ (the mean attacking time is $1/ES_a$) when a is targeting a vulnerability in state j and $\xi_{a,j} = +\infty$ otherwise. Here, ES_a refers to the *exploitability score* of the vulnerability targeted by attack method a. The ES score of a vulnerability is a value between 0 and 10 and a higher ES score means it is easier to exploit the vulnerability [17]. For each vulnerability, we generated 1000 samples from the corresponding exponential distribution and used their average as the attacking time. We used the same migration cost setting as the spatial decision experiment with the migration cost matrix shown in Figure 3.

4.2.2 Results: When the defender is able to decide when to migrate, all three models produce lower cost than the respective models with a fixed unit defending period (see Figure 4 and Figure 5). When α is small, the attacking cost has a major impact on the defender's cost. The shorter defending periods lead to more frequent switches that can efficiently increases the uncertainty of the attacker, thus reduce the attacking cost in the end. As α grows, the migration cost becomes the major factor and the defender's spatial decision plays a major role on the cost. Thus, all the three models set $\tau = 2.6$ eventually to decrease the unit time migration cost. With temporal decisions, BSG and URS are capable to adjust the frequency of migration, which improves their performance significantly. Compared with them, the improvement of MSG is less

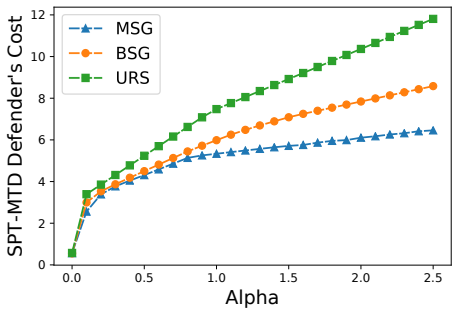

Figure 5: A comparison of the defender's cost in the three spatial-temporal policies - MSG ($\epsilon = 0.1$), BSG and URS as the parameter α increases. $\xi_{a,j} \sim Exp(ES_a)$.

significant. This is because the simulation considers a relatively small updating cost and MSG allows the defender to partially adjust the migration frequency *implicitly* by changing the value of P_{ii} even without introducing explicit temporal decisions. The detailed strategies are provided in our technical report [13].

We observed that, the defender in the MSG model tends to move to configurations with lower attacking cost, namely (PHP, $postgreSQL$) and ($Python$, $postgreSQL$), and never moves out of them. That is, these two configurations are recurrent states (once the system moves in, it can only move between them) and the other two configurations are transient states of the corresponding Markov chain. Essentially, MSG has advantages over BSG in all the scenarios because it allows a more refined trade-off between migration cost and attacking cost.

5 CONCLUSIONS

In this paper we consider a defender's optimal moving target problem in which both the sequence of system configurations and the timing of switching are important. We introduce a Markov Stackelberg Game framework to model the defender's spatial and temporal decision making that aims to minimize the loss caused by compromises of the system and the cost required for migration. We formulate the defender's optimization problem as an average-cost SMDP and transform the SMDP problem into a DTMDP problem that can simplify the problem. Then we propose a value iteration algorithm with convergence guarantee to compute the near-optimal defense policies. Experimental results on real-world data demonstrate that our algorithm outperforms the state-of-the-art benchmarks for MTD. Our Markov Stackellberg Game model precisely captures the defender's spatial-temporal decision making in face of adaptive and sophisticated attackers and can potentially be applied to other security scenarios beyond moving target defense.

ACKNOWLEDGMENTS

This work has been funded in part by NSF grant CNS-1816495. We thank the anonymous reviewers for their constructive comments. We would like to thank Sailik Sengupta for kindly providing their MIQP code for solving BSG.

REFERENCES

[1] John Bather. 1973. Optimal decision procedures for finite Markov chains. Part II: Communicating systems. *Advances in Applied Probability* 5, 3 (1973), 521–540.

[2] Dimitri P Bertsekas. 2012. *Dynamic Programming and Optimal Control (4th edition)*. Vol. 1. Athena scientific Belmont, MA.

[3] Michele Breton, Abderrahmane Alj, and Alain Haurie. 1988. Sequential Stackelberg equilibria in two-person games. *Journal of Optimization Theory and Applications* 59, 1 (1988), 71–97.

[4] Ankur Chowdhary, Adel Alshamrani, Dijiang Huang, and Hongbin Liang. 2018. MTD analysis and evaluation framework in software defined network (MASON). In *Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*. ACM, 43–48.

[5] David Evans, Anh Nguyen-Tuong, and John Knight. 2011. Effectiveness of moving target defenses. In *Moving Target Defense*. Springer, 29–48.

[6] Xiaotao Feng, Zizhan Zheng, Prasant Mohapatra, and Derya Cansever. 2017. A stackelberg game and markov modeling of moving target defense. In *International Conference on Decision and Game Theory for Security*. Springer, 315–335.

[7] Jin B Hong and Dong Seong Kim. 2015. Assessing the effectiveness of moving target defenses using security models. *IEEE Transactions on Dependable and Secure Computing* 13, 2 (2015), 163–177.

[8] Sushil Jajodia, Anup K Ghosh, Vipin Swarup, Cliff Wang, and X Sean Wang. 2011. *Moving target defense: creating asymmetric uncertainty for cyber threats*. Vol. 54. Springer Science & Business Media.

[9] Sushil Jajodia, Noseong Park, Edoardo Serra, and VS Subrahmanian. 2018. Share: A stackelberg honey-based adversarial reasoning engine. *ACM Transactions on Internet Technology (TOIT)* 18, 3 (2018), 30.

[10] Quan Jia, Huangxin Wang, Dan Fleck, Fei Li, Angelos Stavrou, and Walter Powell. 2014. Catch me if you can: A cloud-enabled DDoS defense. In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 264–275.

[11] Liu Jianyong and Zhao Xiaobo. 2004. On average reward semi-Markov decision processes with a general multichain structure. *Mathematics of Operations Research* 29, 2 (2004), 339–352.

[12] Dmytro Korzhyk, Zhengyu Yin, Christopher Kiekintveld, Vincent Conitzer, and Milind Tambe. 2011. Stackelberg vs. Nash in security games: An extended investigation of interchangeability, equivalence, and uniqueness. *Journal of Artificial Intelligence Research* 41 (2011), 297–327.

[13] Henger Li, Wen Shen, and Zizhan Zheng. 2020. Spatial-Temporal Moving Target Defense: A Markov Stackelberg Game Model. *arXiv preprint arXiv:2002.10390* (2020).

[14] Henger Li and Zizhan Zheng. 2019. Optimal Timing of Moving Target Defense: A Stackelberg Game Model. In *Proceedings of the 2019 IEEE Military Communications Conference (MILCOM)*. IEEE.

[15] Pratyusa K Manadhata. 2013. Game theoretic approaches to attack surface shifting. In *Moving Target Defense II*. Springer, 1–13.

[16] Peter Mell, Karen Scarfone, and Sasha Romanosky. 2006. Common vulnerability scoring system. *IEEE Security & Privacy* 4, 6 (2006), 85–89.

[17] Peter Mell, Karen Scarfone, and Sasha Romanosky. 2007. A complete guide to the common vulnerability scoring system version 2.0. In *Published by FIRST-Forum of Incident Response and Security Teams*, Vol. 1. 23.

[18] Thanh Hong Nguyen, Debarun Kar, Matthew Brown, Arunesh Sinha, Albert Xin Jiang, and Milind Tambe. 2016. Towards a science of security games. In *Mathematical Sciences with Multidisciplinary Applications*. Springer, 347–381.

[19] Hamed Okhravi, William W Streilein, and Kevin S Bauer. 2015. *Moving Target Techniques: Leveraging Uncertainty for CyberDefense*. Technical Report. MIT Lincoln Laboratory Lexington United States.

[20] Patrick D O'Reilly. 2009. National vulnerability database (NVD). (2009).

[21] Praveen Paruchuri, Jonathan P Pearce, Janusz Marecki, Milind Tambe, Fernando Ordonez, and Sarit Kraus. 2008. Playing games for security: An efficient exact algorithm for solving Bayesian Stackelberg games. In *Proceedings of AAMAS*. IFAAMAS, 895–902.

[22] Wei Peng, Feng Li, Chin-Tser Huang, and Xukai Zou. 2014. A moving-target defense strategy for cloud-based services with heterogeneous and dynamic attack surfaces. In *2014 IEEE International Conference on Communications (ICC)*. IEEE, 804–809.

[23] Martin L Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons.

[24] Paul J Schweitzer. 1971. Iterative solution of the functional equations of undiscounted Markov renewal programming. *J. Math. Anal. Appl.* 34, 3 (1971), 495–501.

[25] Sailik Sengupta, Ankur Chowdhary, Dijiang Huang, and Subbarao Kambhampati. 2018. Moving target defense for the placement of intrusion detection systems in the cloud. In *International Conference on Decision and Game Theory for Security*. Springer, 326–345.

[26] Sailik Sengupta, Ankur Chowdhary, Abdulhakim Sabur, Dijiang Huang, Adel Alshamrani, and Subbarao Kambhampati. 2019. A Survey of Moving Target Defenses for Network Security. *arXiv preprint arXiv:1905.00964* (2019).

[27] Sailik Sengupta, Satya Gautam Vadlamudi, Subbarao Kambhampati, Adam Doupé, Ziming Zhao, Marthony Taguinod, and Gail-Joon Ahn. 2017. A game theoretic approach to strategy generation for moving target defense in web applications. In *Proceedings of AAMAS*. IFAAMAS, 178–186.

[28] Arunesh Sinha, Fei Fang, Bo An, Christopher Kiekintveld, and Milind Tambe. 2018. Stackelberg Security Games: Looking Beyond a Decade of Success.. In *IJCAI*. 5494–5501.

[29] Aditya K Sood and Richard J Enbody. 2012. Targeted cyberattacks: a superset of advanced persistent threats. *IEEE security & privacy* 11, 1 (2012), 54–61.

[30] Marthony Taguinod, Adam Doupé, Ziming Zhao, and Gail-Joon Ahn. 2015. Toward a moving target defense for web applications. In *2015 IEEE International Conference on Information Reuse and Integration*. IEEE, 510–517.

[31] Satya Gautam Vadlamudi, Sailik Sengupta, Marthony Taguinod, Ziming Zhao, Adam Doupé, Gail-Joon Ahn, and Subbarao Kambhampati. 2016. Moving target defense for web applications using bayesian stackelberg games. In *Proceedings of AAMAS*. IFAAMAS, 1377–1378.

[32] Shardul Vikram, Chao Yang, and Guofei Gu. 2013. Nomad: Towards non-intrusive moving-target defense against web bots. In *2013 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 55–63.

[33] Heinrich Von Stackelberg. 2010. *Market structure and equilibrium*. Springer Science & Business Media.

[34] Mengmeng Yu and Seung Ho Hong. 2015. A real-time demand-response algorithm for smart grids: A stackelberg game approach. *IEEE Transactions on Smart Grid* 7, 2 (2015), 879–888.

[35] Jin Zhang and Qian Zhang. 2009. Stackelberg game for utility-based cooperative cognitiveradio networks. In *Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*. ACM, 23–32.

[36] Quanyan Zhu and Tamer Başar. 2013. Game-theoretic approach to feedback-driven multi-stage moving target defense. In *International Conference on Decision and Game Theory for Security*. Springer, 246–263.

[37] Rui Zhuang. 2015. *A theory for understanding and quantifying moving target defense*. Ph.D. Dissertation. Kansas State University.

[38] Rui Zhuang, Scott A DeLoach, and Xinming Ou. 2014. Towards a theory of moving target defense. In *Proceedings of the First ACM Workshop on Moving Target Defense*. ACM, 31–40.

[39] Rui Zhuang, Su Zhang, Scott A DeLoach, Xinming Ou, and Anoop Singhal. 2012. Simulation-based approaches to studying effectiveness of moving-target network defense. In *National symposium on moving target research*, Vol. 246.