Robust Moving Target Defense against Unknown Attacks: A Meta-Reinforcement Learning Approach

Henger Li and Zizhan Zheng

Tulane University, New Orleans LA 70118, USA
{hli30,zzheng3}@tulane.edu

Abstract. Moving target defense (MTD) provides a systematic framework to achieving proactive defense in the presence of advanced and stealthy attacks. To obtain robust MTD in the face of unknown attack strategies, a promising approach is to model the sequential attacker-defender interactions as a two-player Markov game, and formulate the defender's problem as finding the Stackelberg equilibrium (or a variant of it) with the defender and the leader and the attacker as the follower. To solve the game, however, existing approaches typically assume that the attacker type (including its physical, cognitive, and computational abilities and constraints) is known or is sampled from a known distribution. The former rarely holds in practice as the initial guess about the attacker type is often inaccurate, while the latter leads to suboptimal solutions even when there is no distribution shift between when the MTD policy is trained and when it is applied. On the other hand, it is often infeasible to collect enough samples covering various attack scenarios on the fly in security-sensitive domains. To address this dilemma, we propose a two-stage meta-reinforcement learning based MTD framework in this work. At the training stage, a meta-MTD policy is learned using experiences sampled from a set of possible attacks. At the test stage, the meta-policy is quickly adapted against a real attack using a small number of samples. We show that our two-stage MTD defense obtains superb performance in the face of uncertain/unknown attacker type and attack behavior.

1 Introduction

The relatively static nature of the current IT and infrastructure systems provides adaptive and stealthy cyber-attackers enough time to explore and then exploit a well-designed attack in a "low-and-slow" way [6]. Even worse, the increasingly more complex software technologies make the completely secure defense nearly impossible against an advanced adversary [21]. To reduce or reverse the attacker's asymmetric information advantage, a promising approach is moving target defense (MTD), where the defender proactively updates the system configuration to increase the uncertainty and complexity for potential attackers. MTD has been successfully applied to many technology domains, including web applications [38], cloud computing [30], operating systems [40], and Internet of things [32].

In order to capture the trade-off between security and efficiency in MTD, a game-theoretic approach is often adopted. In particular, early works have modeled the sequential attacker-defender interactions in MTD as a symmetric two-player Markov game [11] or a repeated Bayesian Stackelberg game (BSG) [34]. To achieve robust MTD in the face of uncertain attack behavior, a promising direction is to consider an asymmetric Markov game [22] where the defender (as the leader) first commits to an MTD policy assuming that the attacker (as the follower) will respond to it optimally. Several recent works have followed this direction by formulating the defender's problem as finding the Stackelberg equilibrium (or some variant of it) of the Markov MTD game, using either model-based [23] or model-free [33] reinforcement learning algorithms. The main advantage of this approach is that it provides a guaranteed level of protection by considering the worst-case attack behavior. However, the solution thus obtained can be conservative when the real attack is "weaker" than the worst-case scenario.

Although existing approaches have (partially) addressed the problem of uncertain attack behavior by using a game-theoretic solution concept, they typically assume that the attacker type (including its physical, cognitive, and computational abilities and constraints) is known or sampled from a known distribution. The former rarely holds in practice as the initial guess about the attacker type is often inaccurate, while the latter can lead to overly conservative solutions even when there is no distribution shift (on the attacker type) between when the MTD policy is trained and when it is applied. One possible solution is to consider a fully online approach where the defender assumes zero prior knowledge of the attacker and continuously adapts its policy using feedback obtained during its interactions with the attacker. However, this approach requires collecting a large number of samples covering various attack scenarios, which is typically infeasible in security-sensitive domains.

In this work, we take a first step towards solving the above dilemma, by proposing a two-stage meta-reinforcement learning (meta-RL) based MTD framework. At the training stage, a meta-MTD policy is learned by solving multiple Stackelberg Markov games using experiences sampled from a set of possible attacks. When facing a real attacker with initially uncertain/unknown type and behavior at the test stage, the meta-policy is quickly adapted using a small number of samples collected on the fly. Note that our approach assumes that the defender has a rough estimate of possible attacks, which is weaker than assuming a pre-defined attacker type distribution as in [33]. Further, the meta-defense is still effective even when the real attack at test time is not in the training set, thanks to the generalization property of meta-learning [12]. We show that our new MTD defense obtains superb performance in the practical setting where the defender has very limited prior knowledge of the attacker's type and behavior.

The main contributions of the paper are summarized below.

- We propose a two-stage meta-RL based defense framework for achieving robust moving target defense in the face of uncertain/unknown attack type and behavior.
- We show that the meta-RL defense framework can be efficiently implemented by proving that in our two-player MTD game, the problem of finding the strong Stackelberg equilibrium (SSE) can be reduced to solving a single-agent Markov decision process for the defender.
- Using data collected from the National Vulnerability Database (NVD), we show that our two-stage defense obtains superb performance by quickly adapting the pre-trained meta-defense policy to real attacks. Code is available at https://github.com/HengerLi/meta-RL.

2 The MTD Game Model

In this section, we first describe our system and attack models in detail. We then formulate the attack-defender interactions as a two-player Markov game. Finally, we provide an overview of the proposed two-stage MTD defense framework.

2.1 System model

We consider a time-slotted system where in each time step, the system can be in any one of the n possible configurations. Let s^t denote the configuration of the system in time t. Each configuration consists of multiple adjustable parameters across different layers of the system called adaptation aspects [8]. Typical examples of adaptation aspects include port numbers [25], IP addresses [1, 20, 35], virtual machines [46], operating systems [40], and software programs [19]. We define the system configuration space as $S = [n]$, where n is the number of possible configurations.

At the beginning of each time t, the defender chooses the next system configuration s^t according to a migration policy π_D^t (to be defined). The system stays in the current configuration if $s^t = s^{t-1}$. To increase the attacker's uncertainty, the defense policy should be randomized. Further, the optimal defense policy is in general time-varying and can depend on the system state and the defender's knowledge of the attacker. We assume that a migration happens instantaneously subject to a cost $m_{ij} \geq 0$ when the system moves from configuration i to configuration j. Although not required in our model, we typically have $m_{ii} = 0$. Let $M = \{m_{ij}\}_{n \times n}$ denote the migration cost matrix.

In addition to the migration cost, the defender incurs a loss $l_{s_t} \geq 0$ at the end of time slot t if the system is compromised at t. The value of l_{s_t} varies over the system configuration and the attack type as we discuss below. In this work, we assume that the defender discovers whether the system is compromised or not at the end of each time step t and recovers the system if it is compromised. Therefore, l_{s_t} includes the cost to recover the system from potential damages. This also implies that the system is always protected at the beginning of any time step. Although we consider the simplified setting where the defender receives immediate feedback on potential attacks in this work, our approach can be generalized to the setting when the feedback is delayed or imperfect.

With the above assumptions, it is natural to consider a randomized stationary policy $\pi_{\mathcal{D}} : S \to \triangle(S)$ where $\triangle(S)$ denotes the space of probability distributions over S. Equivalently, $\pi_{\mathcal{D}} = \{\mathbf{p}_i\}_{i \in [n]}$ where p_{ij} gives the probability of moving to configuration j when the system is in configuration i in the previous time step. That is, the defense policy in time t is determined by the system configuration s^{t-1} only. The defender's goal is to minimize its expected total loss including the loss from attacks and the migration cost.

2.2 Threat model

We consider a persistent adversary that continuously attacks the system according to a chosen policy. At the beginning of each time step t, the attacker chooses a system configuration $\tilde{s}^t$ to attack. The attack fails if the attacker chooses the wrong target, that is, $\tilde{s}^t \neq s^t$. Otherwise, the attack succeeds with a probability μ_{s^t} and fails with a probability $1 - \mu_{s^t}$. If the attack succeeds, it leads to a loss of l_{s^t} to the defender (including the recovery cost as discussed above). That is, we model the attacker's *type* using a tuple $(\mu, \mathbf{l})$, where $\mu = \{\mu_j\}_n$ are the attack success rates over the set of configurations and $\mathbf{l} = \{l_j\}_n$ are the unit time system losses for all configurations. The attacker type captures its capability and effectiveness to compromise a set of configurations. Different types of attacks may target the same vulnerability of a configuration but may result in different loss to the system. In practice, these values can be derived from real measurements or publicly available databases such as the National Vulnerability Database (NVD) [5] (see the experiment section for the details).

We assume that the attacker always learns the system configuration s^t at the end of time step t whether the system is compromised at t or not (a worst-case scenario from the defender's perspective). Then it is without loss of generality to consider a randomized stationary policy for the attacker $\pi_{\mathcal{A}} : S \to \triangle(S)$, which can equivalently be defined by $\{\mathbf{q}_i\}_{i \in [n]}$ where q_{ij} denotes the probability of attacking configuration j if the system is in configuration i in the previous time step.

We define an attack as $\xi = (\mu, \mathbf{l}, \pi_A)$ to include its type and policy. In general, the true attack encountered at any time is initially unknown to the defender. However, the defender may have a rough estimate of the possible attacks.

2.3 The Markov game model for MTD

With the definitions and assumptions given above, we can model the sequential interactions between the defender and the attacker of a given type as a two-player general-sum Markov game (MG), denoted by $G = (S, A, \mathcal{P}, r, \gamma)$, where

- S is the state space. In this work, we model the system state s^t at any time t as its configuration. Note that both the defender and the attacker know the true system configuration s^{t-1} at the beginning of time t. Thus, they share the same state space.

- $A = A_{\mathcal{D}} \times A_{\mathcal{A}}$ is the joint action space, where $A_{\mathcal{D}}$ and $A_{\mathcal{A}}$ are the defender's action space and the attacker's action space, respectively. In this work, we have $A_{\mathcal{D}} = A_{\mathcal{A}} = S$. At the beginning of each time step, the defender pick the next configuration s^t to switch to while the attacker picks a target configuration $\tilde{s}^t$ to attack simultaneously according to their policies to fight for control of the system. Let $a^t = (s^t, \tilde{s}^t)$ denote the joint action.
- $\mathcal{P} : S \times A \to \triangle(S)$ is the state transition function that represents the probability of reaching a state $s' \in S$ given the current state $s \in S$ and the joint action $a \in A$. In our setting, the system transition is deterministic as the next state is completely determined by the defender's action.
- $r = \{r_{\mathcal{D}}, r_{\mathcal{A}}\}$ where $r_{\mathcal{D}} : S \times A \to \mathbb{R}_{\leq 0}$ and $r_{\mathcal{A}} : S \times A \to \mathbb{R}_{\geq 0}$ are the reward functions for the defender and the attacker, respectively. Precisely, given the previous system configuration s^{t-1}, the defender's action s^t, and the attacker's action $\tilde{s}^t$, the defender obtains a reward $r^t_{\mathcal{D}} = r_{\mathcal{D}}(s^{t-1}, (s^t, \tilde{s}^t)) = -\mathbf{1}_{s^t = \tilde{s}^t} \mu_{s^t} l_{s^t} - m_{s^{t-1} s^t}$ and the attack obtains a reward $r^t_{\mathcal{A}} = r_{\mathcal{A}}(s^{t-1}, (s^t, \tilde{s}^t)) = \mathbf{1}_{s^t = \tilde{s}^t} \mu_{s^t} l_{s^t}$ where $\mathbf{1}_{(.)}$ is the indicator function.
- $\gamma \in (0, 1]$ is the discount factor.

Given the initial state s^0 and a pair of policies $\pi_{\mathcal{D}}$ and $\pi_{\mathcal{A}}$ for the defender and the attacker, respectively, let $V_i^{\pi_{\mathcal{D}}, \pi_{\mathcal{A}}}(s^0) = \mathbb{E}_{\pi_{\mathcal{D}}, \pi_{\mathcal{A}}}[\sum_{t=0}^{\infty} \gamma^t r_i(s^{t-1}, (s^t, \tilde{s}^t) | s^0)]$ denote the total expected return for the player i, where $i \in \{\mathcal{D}, \mathcal{A}\}$. The goal of each player is to maximize its total expected return. Similar to normal-form games, various solution concepts have been considered for Markov games, including Nash equilibrium [17], correlated equilibrium [45], and Stackelberg equilibrium [22]. In this work, we consider the Stackelberg equilibrium as the solution concept. Given the asymmetric information structure commonly seen in cybersecurity and to derive a robust defense, it is typical to consider the Stackelberg equilibrium (or a variant of it) with the defender as the leader and the attacker as the follower. In particular, the defender first commits to a policy π_D, and the attacker as the follower observes π_D (e.g., by stealthily collecting enough samples of defense actions before attacking) and then picks π_A optimally. This approach has been extensively studied for one-shot security games including models such as Stackelberg security games (SSG) and Bayesian Stackelberg games (BSG) [28, 36]. Recently, it has been applied for defending against persistent attacks using techniques such as MTD [23, 34, 33]. Note that when π_D is random as is typical in security domains, knowing π_D does not let the attacker learn the defender's true action in each time step.

When the attacker has multiple best responses to the defender's policy, a common assumption in security games is that the attacker always chooses the one in favor of the defender. This gives the concept of the Strong Stackelberg equilibrium (SSE) formally defined below, which is the solution concept we use in this work.

Definition 1. ***(SSE)*** *For each defense policy $\pi_{\mathcal{D}}$, let $B(\pi_{\mathcal{D}})$ denote the set of attack policies that maximize $V_{\mathcal{D}}^{\pi_{\mathcal{D}},\cdot}(s^0)$ for any s^0, i.e.,*

$$B(\pi_{\mathcal{D}}) := \{\pi_{\mathcal{A}} : V_{\mathcal{A}}^{\pi_{\mathcal{D}},\pi_{\mathcal{A}}}(s^0) = \max_{\pi'_{\mathcal{A}}} V_{\mathcal{A}}^{\pi_{\mathcal{D}},\pi'_{\mathcal{A}}}(s^0), \forall s^0 \in S\} \tag{1}$$

A pair of stationary policies $(\pi^*_{\mathcal{D}}, \pi^*_{\mathcal{A}})$ forms a strong Stackelberg equilibrium if for any $s^0 \in S$, we have

$$V_{\mathcal{D}}^{\pi^*_{\mathcal{D}},\pi^*_{\mathcal{A}}}(s^0) = \max_{\pi_{\mathcal{D}},\pi_{\mathcal{A}} \in B(\pi_{\mathcal{D}})} V_{\mathcal{D}}^{\pi_{\mathcal{D}},\pi_{\mathcal{A}}}(s^0) \tag{2}$$

Since the Markov game defined above has finite state and action spaces and the transition is deterministic, an SSE (when the leader is restricted to randomized stationary policies) is guaranteed to exist and provides a unique game value to the defender [37, 3, 41]. Further, an SSE can be found using either an exact solution [41] or an approximation solution such as Stackelberg Q-learning [22, 33] and Stackelberg policy gradient [42, 18] for large games. The following simple observation justifies the use of SSE as the solution concept in our MTD framework. In particular, it shows that by following the defense policy given by an SSE, the defender's loss in the face of an arbitrary attacker is upper bounded by the loss specified by the SSE.

Lemma 1. *Let $(\pi^*_{\mathcal{D}}, \pi^*_{\mathcal{A}})$ be a strong Stackelberg equilibrium. For an arbitrary attack policy $\pi_{\mathcal{A}}$, we have $V_{\mathcal{D}}^{\pi^*_{\mathcal{D}},\pi_{\mathcal{A}}}(s^0) \geq V_{\mathcal{D}}^{\pi^*_{\mathcal{D}},\pi^*_{\mathcal{A}}}(s^0)$ for any s^0.*

Proof. For any pair of policies $(\pi_{\mathcal{D}}, \pi_{\mathcal{A}})$ and initial state s^0, we can write $V_{\mathcal{D}}^{\pi_{\mathcal{D}},\pi_{\mathcal{A}}}(s^0) = -L(\pi_{\mathcal{D}}, \pi_{\mathcal{A}}) - \mathcal{M}(\pi_{\mathcal{D}})$, where the first term captures the total expected attack loss, and the second term specifies the total expected migration cost. The result then follows by observing that $V_{\mathcal{D}}^{\pi^*_{\mathcal{D}},\pi_{\mathcal{A}}}(s^0) = -L(\pi^*_{\mathcal{D}}, \pi_{\mathcal{A}}) - \mathcal{M}(\pi^*_{\mathcal{D}}) \geq -L(\pi^*_{\mathcal{D}}, \pi^*_{\mathcal{A}}) - \mathcal{M}(\pi^*_{\mathcal{D}}) = V_{\mathcal{D}}^{\pi^*_{\mathcal{D}},\pi^*_{\mathcal{A}}}(s^0)$, where the inequality is due to the fact that $\pi^*_{\mathcal{A}}$ is the attacker's best response to $\pi^*_{\mathcal{D}}$ and the migration cost is independent of the attack policy.

Note that Lemma 1 does not hold for general non-zero-sum Markov games. It holds in our setting because the defender's total attack loss $L(\pi_{\mathcal{D}}, \pi_{\mathcal{A}})$ is exactly the attacker's total gain and the migration cost $\mathcal{M}(\pi_{\mathcal{D}})$ is independent of the attacker's policy.

2.4 Two-stage defense overview

The asymmetric Markov game presented above provides a reasonable solution to robust MTD by considering the worst-case attack behavior. However, the solution thus obtained can be overly conservative in the face of a "weaker" attacker (e.g., a dumb attacker that does not respond to the defense strategically). Moreover, to solve the game, the defender needs to know the exact attacker type, which can lead to a poor solution when the true attacker type deviates from

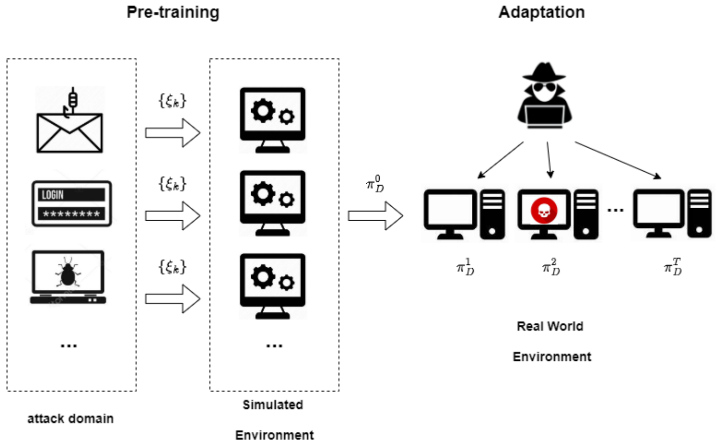


Fig. 1: Two-stage defense overview.

the guessed one as shown in our experiments. One possible solution is to consider a distribution of possible attacker types and optimize the defense policy for either the worst case using distributionally robust optimization [9] or the average case using a Bayesian approach [16, 33]. However, these approaches miss the opportunity of online adaption and can lead to suboptimal defense.

To effectively thwart a potentially unknown attacker, we propose a meta-learning based two-stage defense framework (see Figure 1) to pre-train a meta-policy on a variety of attacks in an simulated environment, such that it can be quickly adapted to a new attack using only a small number of real samples.

The training stage is implemented in a simulated environment, which allows sufficient training using trajectories generated from a pool of potential attacks. The possible set of attack types can be generated using existing databases, such as the National Vulnerability Database (NVD) [5] or penetration testing tools like Kali Linux [2]. On the other hand, to generate diverse attack behavior, we consider both the worst-case scenario specified by the SSE, where the attacker responds to the defense policy optimally, as well as “weaker” attacks, e.g., a random attack that is agnostic of the defense policy. At the test stage, the learned meta-policy π_D^0 is applied and updated using feedback (i.e., rewards) received in the face of real attacks that are not necessarily in the training set (in terms of both attack type and attack behavior).

3 Meta-RL based MTD Solution Framework

In this section, we discuss the details of our meta-reinforcement learning based MTD solution. We first present an important observation that from the defender’s perspective, the problem of finding its optimal MTD policy can be reformulated as a single-agent Markov decision process (MDP). This result holds both when the attacker follows a fixed stationary policy that is known to the defender as well as when it first observes the defense policy and then responds to it optimally (as in the case of solving the SSE of the Markov game presented above). This observation allows us to reduce the bilevel optimization problem to a single level optimization problem. Based on this observation, we then present our two-stage defense that adapts model-agnostic meta-learning (MAML) [14, 27] to MTD.

3.1 Reducing the MG to an MDP

Before starting the main results in this section, we first generalize the definition of SSE by allowing the attacker to respond to the defense policy in a suboptimal way, which allows us to incorporate diverse attack behavior into meta-learning. In particular, we will assume that after the defender commits to a stationary policy π_D, the attacker chooses a policy $\mathbf{q}_s \in R(s, \pi_D)$ for any s, where $R(\cdot, \pi_D)$ denotes a set of response policies [22]. Note that this includes the cases when the attacker responds in an optimal way (as in the case of SSE), when it responds in a suboptimal way, as well as the case when the attack policy is fixed and independent of the defense. We define a pair of policies (π_D, π_A) to be a *generalized SSE* if π_D minimizes the defender's loss assuming the attack responds according to its response set R and in favor of the defender when there is a tie.

We first show that the defender's problem can be viewed as a single agent MDP whenever the following assumptions hold.

Assumption 1. *For any state $s \in S$, the attacker's response set is either a singleton or all the responses are equally good to the defender.*

Assumption 2. *For any state $s \in S$, the attacker's policy $\mathbf{q}_s$ in state s only depends on s and $\mathbf{p}_s$, i.e., the defender's policy at state s, and is independent of the defender's policy in other states. That is, given π_D, we can write $\mathbf{q}_s \triangleq R(s, \mathbf{p}_s)$ for any $s \in S$.*

In particular, Assumption 1 allows the defender to infer the attack policy under each state, while Assumption 2 ensures that the defender's reward at any time depends on the current state and defense action only but not the future states (see the proof of Lemma 2).

Lemma 2. *When Assumptions 1 and 2 hold, the optimal defense policy in the sense of a generalized SSE can be found by solving a single-agent Markov decision process with continuous actions for the defender.*

Proof. Consider the following MDP for the defender $M = (S, A', T', r', \gamma)$, which is derived from the Markov game in Section 2.3, where

- S is the state space, which is defined as the set of configurations.
- $A' = \triangle(S)$ is the action space of the defender, where we redefine the defender's action in configuration s as the probability vector $\mathbf{p}_s$, which is critical for converting the Markov game into an MDP.

- $\mathcal{P}' : S \times A' \to \triangle(S)$ is the state transition function. Given the previous system configuration s and the defender's action $\mathbf{p}_s$, the probability of reaching a configuration s' in the current time step is $\mathcal{P}'(s'|s, \mathbf{p}_s) = p_{ss'}$. Note that the state transition is stochastic rather than deterministic as in the Markov game.
- $r' : S \times A' \to \mathbb{R}_{\leq 0}$ is the reward function for the defender. Given the previous configuration s^{t-1}, the defender's action $\mathbf{p}_{s^{t-1}}$, the attacker's policy can be represented as $\mathbf{q}_{s^{t-1}} = R(s^{t-1}, \mathbf{p}_{s^{t-1}})$ according to Assumptions 1 and 2 (with ties broken arbitrarily when $R(s^{t-1}, \mathbf{p}_{s^{t-1}})$ is not a singleton). The defender's reward is defined as $r'(s^{t-1}, \mathbf{p}_{s^{t-1}}) = \sum_{s^t, \tilde{s}^t} p_{s^{t-1}s^t} q_{s^{t-1}\tilde{s}^t} r_D(s^{t-1}, (s^t, \tilde{s}^t))$.
- $\gamma \in (0, 1)$ is the discount factor, which is the same as the discount factor in the Markov game.

It is crucial to note that while the defender's reward function r_D in the Markov game depends on the joint action of both the attacker and the defender, r' only depends on the defender's action and is therefore well defined. This is achieved by redefining the defection's action in any configuration as the corresponding migration probability vector and using the two assumptions. Given the MDP above, it is easy to check that the problem of finding the best defense policy can be reduced to finding a deterministic policy $\pi : S \to A'$ that maximizes the total expected return $\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r'(s^{t-1}, \mathbf{p}_{s^{t-1}}|s^0)]$ in the MDP.

Note that both assumptions automatically hold when the attacker is agnostic to the defense policy, and Assumption 1 holds for an SSE. Below we show that for any stationary defense policy, there is a best-response attack that satisfies Assumption 2.

Lemma 3. *For any stationary defense policy* $\mathbf{p}$*, any deterministic policy* $\mathbf{q}$ *of the following form is in the attacker's best response set: for any* $i \in S$*, there is* $j \in S$ *such that* $q_{ij} = 1$*,* $q_{ik} = 0$ *for* $k \neq j$*, and* $j \in \arg\max_j p_{ij}\mu_j l_j$*.*

Proof. Given a stationary defense policy $\mathbf{p}$ and initial state s^0, the attacker's goal is to maximize its expected total reward, that is

$$\begin{aligned}
\max_{\mathbf{q}} V_{\mathcal{A}}^{\mathbf{p},\mathbf{q}}(s^0) &= \max_{\mathbf{q}} \mathbb{E}_{\mathbf{p},\mathbf{q}} \left(\sum_{t=0}^{\infty} \gamma^t r_{\mathcal{A}}(s^{t-1}, (s^t, \tilde{s}^t)) \right) \\
&\overset{a}{=} \max_{\mathbf{q}(s^0)} \sum_{s^1, \tilde{s}^1} p_{s^0 s^1} q_{s^0 \tilde{s}^1} r_{\mathcal{A}}(s^0, (s^1, \tilde{s}^1)) + \gamma \max_{\mathbf{q}} V_{\mathcal{A}}^{\mathbf{p},\mathbf{q}}(s^1) \\
&= \max_{s^1} p_{s^0 s^1} r_{\mathcal{A}}(s^0, (s^1, s^1)) + \gamma \max_{\mathbf{q}} V_{\mathcal{A}}^{\mathbf{p},\mathbf{q}}(s^1) \\
&\overset{b}{=} \max_{s^1} p_{s^0 s^1} \mu_{s^1} l_{s^1} + \gamma \max_{\mathbf{q}} V_{\mathcal{A}}^{\mathbf{p},\mathbf{q}}(s^1)
\end{aligned}$$

where (a) is due to the fact that the transition probabilities only depend on the current state and the defender's action, and (b) follows from the definition of $r_{\mathcal{A}}$. The result then follows by induction.

The following result follows directly from Lemma 2 and Lemma 3.

Proposition 1. *An SSE defense policy can be found by solving a single-agent Markov decision process with continuous actions for the defender.*

From the above proposition, the defender can identify a near-optimal defense by solving the MDP above whenever the attack type is known and the attack behavior follows a known response set. This can be done using either dynamic programming or a model-free reinforcement learning algorithm with samples generated from a simulator of the MDP, the latter is more suitable for MTDs with a large configuration space. In both cases, the defense policy can be derived in a simulated environment without interacting with the true attacks.

3.2 Robust defense via meta-RL

In order to cope with real attacks with uncertain or unknown types and behaviors, we propose a meta reinforcement learning based two-stage defense framework. The core idea is to pre-train a meta-policy over a distribution of defender's MDPs, where each MDP corresponds to interactions with a particular attack. At test time, the pre-trained meta defense policy is quickly adapted to the true attack encountered using a small number of samples.

Meta-learning (or learning-to-learn) is a principled approach for developing algorithms that can adapt experiences collected from training tasks to solving unseen tasks quickly and efficiently, which has been successfully applied to various learning domains including reinforcement learning. In this work, we adopt a first-order model-agnostic meta-learning algorithm, Reptile [27] to MTD by viewing the defender's problem of solving its MDP against a particular attack $\xi^k = (\mu_k, \mathbf{l}_k, R_k)$ as a task (see Algorithm 1). Here R_k is the response function defined in Section 3.1 that completely captures the attack behavior. The input to the algorithm includes a distribution of attacks $\mathcal{P}(\xi)$, which can be estimated from public datasets or through experiments, and two step sizes.

We consider the defender's policy to be represented by a parametrized function (e.g., a neural network) $\pi_{\mathcal{D}}(\theta)$ with parameters θ. The algorithm starts with an initial model θ^0, and updates it over T iterations. In each iteration, the algorithm first samples a batch of K attacks from $\mathcal{P}(\xi)$. For each attack, a trajectory of length H is generated, which is used to compute the meta-policy θ_k^t for the k-th attack by performing gradient descent for m steps with step size α. Given an initial state s_0, we define the loss function for a particular attack over H time steps as

$$\mathcal{L}_{\xi^k}(\pi_{\mathcal{D}}(\theta)) = -\mathbb{E}_{\pi_{\mathcal{D}}(\theta)}[\sum_{t=1}^{H} r'(s^{t-1}, \mathbf{p}_{s^{t-1}})] \quad (3)$$

We consider a fixed horizon setting (with $\gamma = 1$) such that the defender is allowed to query a limited number of samples for updating its policy. Let H denote the length of an episode. The policy gradient method is performed on the loss $\mathcal{L}_{\xi^k}$ starting with initial parameters $\theta_k^t = \theta^{t-1}$ and returns the final

Algorithm 1 Reptile Meta-Reinforcement Learning for Robust MTD

```
Input: a distribution over attacks $\mathcal{P}(\xi)$, step size parameters $\alpha, \beta$
Output: $\theta^T$
randomly initialize $\theta^0$
for iteration = 1 to $T$ do
  Sample $K$ attacks $\xi^k$ from $\mathcal{P}(\xi)$
  for all $\xi^k$ do
    Sample a trajectory $\mathcal{H}$ of length $H$ using $\pi_{\mathcal{D}}(\theta^{t-1})$ and attack $\xi^k$
    $\theta_k^t \leftarrow \theta^{t-1}$
    for $l = 1$ to $m$ do
      Evaluate $\nabla_\theta \mathcal{L}_{\xi^k}(\pi_{\mathcal{D}}(\theta_k^t))$ using $\mathcal{H}$ and $\mathcal{L}_{\xi^k}$ in Equation (3)
      $\theta_k^t \leftarrow \theta_k^t - \alpha \nabla_\theta \mathcal{L}_{\xi^k}(\pi_{\mathcal{D}}(\theta_k^t))$
    end for
  end for
  Update $\theta^t \leftarrow \theta^{t-1} - \beta \sum_{k=1}^{K}(\theta^{t-1} - \theta_k^t)$
end for
```

parameters. With model parameters collected from all the tasks in the batch, θ^t is then updated towards these new parameters. Note that the single-task gradient is defined as $(\theta - \theta_k)/\alpha$, where α is the step size used by the gradient decent operation. According to [27], we assume that policy $\pi_{\mathcal{D}}(\theta_k)$ achieves the best performance for task ξ^k when θ_k lays on the surface of the manifold of optimal network configuration. We want to find the closest point on the optimal task manifold. This cannot be computed exactly, but Reptile approximates it using $\mathcal{L}_{\xi^k}$. The trained meta-defense policy is then adapted at test time with a few more samples from interactions with the real attacks.

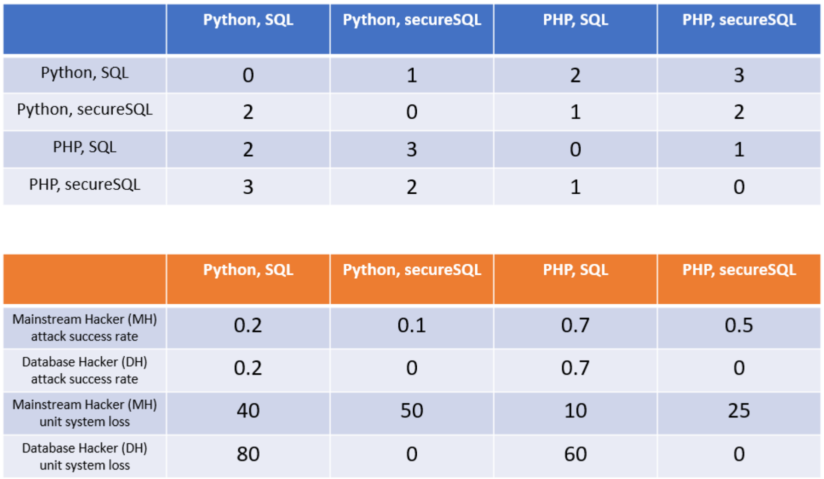
Remark 1. In more complicated scenarios where the defender's SSE policy cannot be formulated as an MDP, we may still adopt the above meta-learning framework by solving the Markov game defined in Section 2.3 to identify the SSE defense at both the meta-training stage and the testing stage. Extension of the meta-RL algorithm to this more general setting (e.g., delayed/noisy feedback from both attacker's and defender's perspectives) is left to future work.

4 Experiment Results

In this section, we validate our MTD framework using the data from the National Vulnerability Database (NVD) [5]. We aim to understand if the SSE defense can provide a robust solution in the face of uncertain/unknown attacks and how meta-learning can help further improve the security of the system.

4.1 Experiment setup and baselines

System configurations. We consider a web system with four configuration $S = \{(Python, SQL), (Python, secureSQL), (PHP, SQL), (PHP, secureSQL)\}$ across



	Python, SQL	Python, secureSQL	PHP, SQL	PHP, secureSQL
Python, SQL	0	1	2	3
Python, secureSQL	2	0	1	2
PHP, SQL	2	3	0	1
PHP, secureSQL	3	2	1	0

	Python, SQL	Python, secureSQL	PHP, SQL	PHP, secureSQL
Mainstream Hacker (MH) attack success rate	0.2	0.1	0.7	0.5
Database Hacker (DH) attack success rate	0.2	0	0.7	0
Mainstream Hacker (MH) unit system loss	40	50	10	25
Database Hacker (DH) unit system loss	80	0	60	0

Fig. 2: Defender's and attacker's parameters. The upper table gives the migration cost for the MTD system. Each row represents a source configuration and each column represents a destination configuration. The lower table shows the attack parameters including the attack success rate and the unit time system loss (see Section 2.2 for details), for the Mainstream Hacker (MH) and the Database Hacker (DH), respectively.

two layers, similar to [34, 23]. The first layer specifies the programming language used for web applications including Python and PHP. The second layer specifies the database technology used, where SQL stands for the case when the database layer naively uses the structured query language without protection, while secureSQL indicates the case when the database layer is protected using methods including but not limited to isolating the database server, regulating SQL traffic, and restricting the ability users to perform unauthorized tasks [39, 29]. By doing so, we create some 'safe' configurations for certain types of attacks (e.g., configurations using secureSQL is immune to the Database Hacker defined below). The migration cost matrix is given in Figure 2, which is designed with the following considerations in mind: (1) switching between configurations across different layers incurs a higher cost than switching within the same layer; (2) the migration cost between two configurations could be asymmetric; (3) the migration cost should be significantly lower than the loss caused by a successful attack.

Attack types. Inspired by [34, 23], we derive the key attack parameters (i.e., the attack success rate and the unit time system loss) from the Common Vulnerabilities Exposure (CVE) scores given by the Common Vulnerability Scoring System (CVSS) [26] in NVD. In particular, for a given vulnerability, an attack with an Impact Score (IS) $\in [0, 10]$ and an Exploitability Scores (ES) $\in [0, 10]$ will generate 10×IS unit time loss, and will have a 0.1×ES attack success rate.

We consider two attack types, the Mainstream Hacker (MH) and the Database Hacker (DH) as in [34]. An MH attack can exploit a large set of vulnerabilities, but causes less loss when the attack succeeds. In contrast, the DH targets only a few database specific vulnerabilities, but causes critical loss when it succeeds. We collect CVEs in NVD ranging from 2019 to 2021 according to CVSS v3.0, targeting keywords Python, PHP, and SQL, then calculate their rounded average scores. The attack parameters obtained for MH and DH are shown in Figure 2. We use them to simulate attack types at test time while considering a broader class of attack types at training time (see Section 4.3).

Baseline defense and attack strategies. We consider the following defense strategies in the experiments.

- Uniform Random Strategy (URS) [36]: In a URS defense, the defender uniformly samples a configuration from S to switch to in each time step.
- Reinforcement Learning Strategy (RL): In an RL defense, the defender identifies its optimal defense by solving a single-agent MDP using reinforcement learning. This requires the defender (as the leader) to guess the attacker's response as discussed in Section 3.1. This includes the SSE defense as a special case when the attacker is assumed to respond optimally.

We consider the following attack strategies in the experiments.

- Uniform Random Strategy (URS) [36]: In a URS attack, the attacker uniformly samples a configuration from S to attack in each time step.
- Reinforcement Learning Strategy (RL): When the defender adopts a stationary policy, the attacker (as the follower) can learn the defense policy and then identify its optimal attack policy by solving a single-agent MDP using reinforcement learning. We call such an attack RL attack.
- Best Response Strategy (BS): Instead of using the RL attack, the attacker can also identify its best response by solving a simple optimizing problem in each time step as proved in Lemma 3. We call this attack BS attack.
- Worst Response Strategy (WS): We further consider the opposite of the best-response attack where the attacker takes the worst-response (WS) action in each time step, that is, $q_{ij} = 1$ for some $j \in \arg\min_j p_{ij}\mu_j l_j$ and $q_{ik} = 0$ for $k \neq j$ (see Lemma 3).

Meta-RL settings. We implement our MTD environment using Pytorch and OpenAI gym [7]. We use Twin Delayed DDPG (TD3) [15] implemented by OpenAI Stable Baseline3 [31] as the policy updating algorithm in both the pre-training and adaptation stages. The initial state is uniformly sampled from the configuration space. At the training stage, we set the number of iterations $T = 100$. In each iteration, we uniformly sample $K = 20$ attacks from the attack domain (see Section 4.3 for details). For each attack, we generate a trajectory of length $H = 100$ and update the corresponding meta-policy for 10 steps using TD3 (i.e., $m = 10$). At the test stage, the meta-policy is adapted for 100 steps using TD3 with $T = 10$, $H = 10$, and $m = 1$. Other parameters are described as follows: single task step size $\alpha = 0.001$, meta-optimization step size $\beta = 1$,

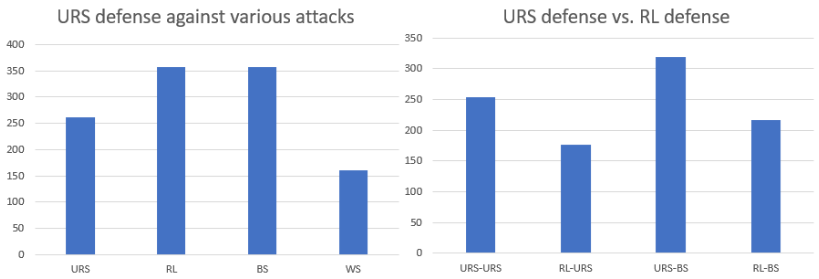


Fig. 3: Left: a comparison of defender's total loss for 100 time steps under the URS defense against different attacks (i.e., URS, RL, BS, and WS). Right: a comparison of defender's total loss for 100 time steps under URS defense and RL defense against URS attack and BS attack, respectively. Here the RL defense is trained against URS attack. All parameters are described in Section 4.1.

adaptation step size = 0.01, the policy model is *MlpPolicy*, batch size = 100 and $\gamma = 1$ for updating the target networks. All the experiments are conducted on the same 2.30GHz Linux machine with 16GB NVIDIA Tesla P100 GPU. We run all the tests for 1,000 times and report the mean value. Since the standard deviations are below 0.04, we omit the error bar for better visualization.

4.2 Results for a single attack type

Before presenting the results for our meta-RL based MTD framework, we first verify our observations made in previous sections regarding the best-response attack and SSE by considering a single attack type (the Mainstream Hacker (MH)) where the defender knows the attack parameters (μ and $\mathbf{l}$) but not necessarily the attack policy.

Optimal attack vs. random attack. Figure 3(left) compares the performance of URS defense under different attacks. Among them, RL attack and BS attack incur the highest total loss (∼356) in 100 steps, indicating that reinforcement learning can also be used to identify the best response attack although it is more time-consuming to train. Both URS and BS attacks perform better than WS as expected. Figure 3(right) shows the performance of URS defense and RL defense (trained against the URS attack) in the face of URS attack and BS attack, respectively. RL against URS (RL-URS) achieves the lowest cost, indicating that the RL defense is effective when the defender knows both the attack type and attack policy. In contrast, RL-BS incurs a higher cost indicating the impact when the defender's guess on the attack policy is wrong. However, RL defense outperforms URS defense in both cases.

The robustness of SSE defense. We show in Lemma 3 that when the defender adopts the SSE defense, it can obtain a guaranteed (albeit conservative) level of protection even when the attacker deviates from the best response behavior. We verify this observation in Figure 4 where under the SSE defense,

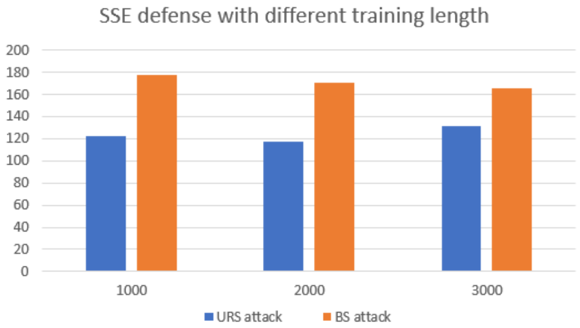


Fig. 4: A comparison of defender's total loss over 100 time steps under the SSE defense (with different training length) against URS and BS attacks.

the total loss incurred by the BS attack is always higher than the URS attack over all the training lengths. We further observe that longer training improves the defense performance against the BS attack (the attack used in training the defense policy), but is not always helpful to defend against other attacks (due to overfitting).

4.3 The effectiveness and efficiency of meta-RL defense

To demonstrate the advantage of our meta-RL defense, we consider two attacks at the test stage, the Mainstream Hacker (MH) and the Database Hacker (DH), both using the BS strategy as the attack policy. We further consider two meta-training settings. In the white-box setting, all attacks for training are sampled from the four combinations of the two attack types (e.g., MH and DH) and the two attack policies (e.g., URS and BS). In the black-box setting, the attack domain includes infinite number of attack types, where each has an uniformly random IS $\in [0, 10]$ and an uniformly random ES $\in [0, 10]$ for every configuration, with the attack policies uniformly sampled from URS and BS. Thus, the white-box setting captures the scenario when the test-stage attack is uncertain to the defender while the black-box setting captures the scenario when the test-stage attack type is essentially unseen to the defender. In both cases, we use the SSE defense trained against the MH attack as the baseline.

Figure 5 shows the defender's test-stage loss over 1,000 time steps, where the meta-policy is adapted for 100 steps at the beginning of the test stage. Note that the RL policy trained against the MH attack is optimal in the face of the same MH attack at test time, which is expected. We observe that both the white-box and the black-box meta-RL defenses perform close to the optimal defense policy. For the DH attack, the RL defense performs poorly due to the mismatch between the training stage and testing stage attack types, while both meta-RL defenses significantly reduce the defender's cost, indicating the benefit of meta-learning. Specifically, the white-box meta-RL defense quickly adapts to

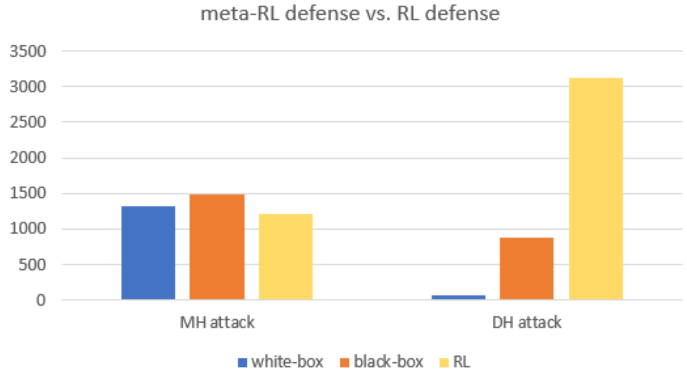


Fig. 5: A comparison of defender's total loss over 1,000 testing time steps for white-box and black-box meta-RL defenses and RL defense (trained against MH attack), against MH attack and DH attack, respectively. The meta-RL polices are adapted for 100 steps at the beginning of the test stage. All attacks adopt the BS strategy at the test stage.

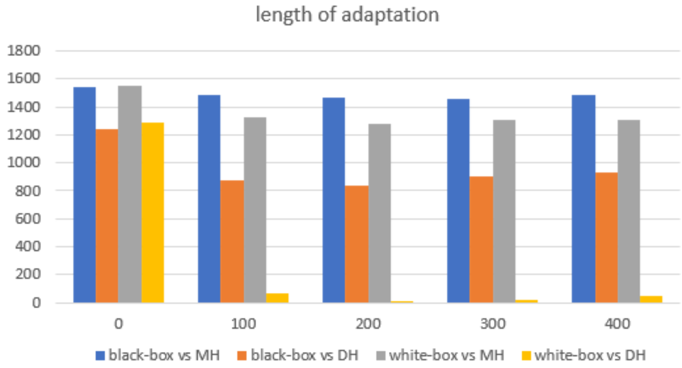


Fig. 6: The defender's total loss over 1,000 testing time steps under different adaptation length for black-box and white-box meta-RL defenses against MH and DH attacks, respectively. We only show the results for policies obtained at step 0, 100, 200, 300 and 400, respectively. All attacks adopt the BS strategy at the test stage.

the optimal policy, i.e., staying at configuration $(Python, secureSQL)$ or configuration $(PHP, secureSQL)$ since DH has zero impact on these states.

Figure 6 shows how the defender's test-stage loss over 1,000 time steps varies across different adaptation duration, where we only plot the results for policies obtained at time step 0, 100, 200, 300, and 400, respectively. As expected, the white-box meta-RL adapts faster since the both attacks considered at testing time are included in the training stage attack domain. Note that the adaptation for the MH attack is less significant since the meta-policy without adaptation is already close to optimal with respect to the MH attack. Although not shown in the figure, we observe that it is much more effective to adapt a well-trained meta-policy (around 10 times faster) than training a new RL policy from scratch (e.g., starting from a random initial policy) to obtain a similar level of protection.

5 Related Work

Stackelberg games for MTD. Stackelberg games [4] have been widely used to derive robust defense against strategic attacks. In particular, one-shot Stackelberg games such as the classic Stackelberg security games (SSG) have been extensively studied for various physical security and cybersecurity domains [36]. In the vanilla SSG, the defender commits to a mixed strategy while the attack observes the strategy and chooses a best response accordingly. Various extensions of SSG have been considered including Bayesian Stackelberg games (BSG) [28], where a Bayesian approach is adopted to model the defender's uncertainty about attack types. A repeated BSG has been applied to MTD in [34] where the defense policy is independent of the current system configuration. More recently, asymmetric Markov game models have been proposed for MTD [13, 24, 23], which allow state-dependent defense but assume a fixed attack type. In [33], a Bayesian Stackelberg Markov game (BSMG) is proposed where the attack type can vary over time according to a pre-defined distribution.

Meta-reinforcement learning. The purpose of meta-RL is to generalize the experience learned from training tasks to new tasks that can be never encountered during training. The adaptation stage in meta-learning is required to have limited exposure to the new tasks, which is crucial for security or safety sensitive domains as it can be expensive or even dangerous to collect samples in real settings. Various approaches have been proposed for meta-learning including metrics-based, model-based, and optimization-based methods [44]. In [43] and [10], the meta-learning algorithm is encoded in the weights of a recurrent neural network, hence gradient descent is not performed at test time. In [14], a model-agnostic meta-learning (MAML) framework is proposed, which does not require a recurrent model, but instead learns the parameters of any standard model via solving a second-order meta-objective optimization. Reptile [27] is a first-order meta-learning optimization algorithm, which is similar to MAML in many ways, given that both relying on meta-optimization through gradient descent and both are model-agnostic.

6 Conclusion

In this paper, we propose a meta-reinforcement learning based moving target defense framework. The key observation of our work is that existing security game models built upon the strong Stackelberg equilibrium (SSE) solution concept (and its Bayesian variant) can lead to suboptimal defense due to the distribution shift between the attacks used for training the defense policy and the true attacks encountered in reality. To this end, we first formulate the MTD problem as an asymmetric Markov game with the defender as the leader and the attacker as the follower. We show that the best-response attack at each state can be determined by the current state of the system and the defender's mix strategy in the current state. This allows us to formulate the problem of finding the SSE defense as a single agent Markov decision process (MDP). We then show that by pre-training the defense policy across a pool of attacks (defined as different MDPs) using model-agnostic meta-learning, the meta-defense policy can quickly adapt to the true attacks at test time. Our two-stage defense approach improves upon the SSE defense in the presence of uncertain/unknown attack type and attack behavior.

Acknowledgements

This work has been funded in part by NSF grant CNS-1816495. We thank the anonymous reviewers for their valuable and constructive comments.

References

1. Al-Shaer, E., Duan, Q., Jafarian, J.H.: Random host mutation for moving target defense. In: International Conference on Security and Privacy in Communication Systems. pp. 310–327. Springer (2012)
2. Allen, L., Heriyanto, T., Ali, S.: Kali Linux–Assuring security by penetration testing. Packt Publishing Ltd (2014)
3. Basar, T.: Lecture notes on non-cooperative game theory. https://www.hamilton.ie/ollie/Downloads/Game.pdf (2010)
4. Başar, T., Olsder, G.J.: Dynamic noncooperative game theory. SIAM (1998)
5. Booth, H., Rike, D., Witte, G.A., et al.: The national vulnerability database (nvd): Overview (2013)
6. Bowers, K.D., Dijk, M.E.V., Juels, A., Oprea, A.M., Rivest, R.L., Triandopoulos, N.: Graph-based approach to deterring persistent security threats. US Patent 8813234 (2014)
7. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: Openai gym. ArXiv **abs/1606.01540** (2016)
8. Cho, J.H., Sharma, D.P., Alavizadeh, H., Yoon, S., Ben-Asher, N., Moore, T.J., Kim, D.S., Lim, H., Nelson, F.F.: Toward proactive, adaptive defense: A survey on moving target defense. IEEE Communications Surveys & Tutorials **22**(1), 709–745 (2020)
9. Derman, E., Mannor, S.: Distributional robustness and regularization in reinforcement learning. In: The Theoretical Foundations of Reinforcement Learning Workshop at ICML 2020 (2020)
10. Duan, Y., Schulman, J., Chen, X., Bartlett, P.L., Sutskever, I., Abbeel, P.: RL^2: Fast reinforcement learning via slow reinforcement learning. arXiv preprint arXiv:1611.02779 (2016)
11. Eldosouky, A., Saad, W., Niyato, D.: Single controller stochastic games for optimized moving target defense. In: IEEE International Conference on Communications (ICC) (2016)

12. Fallah, A., Mokhtari, A., Ozdaglar, A.: Generalization of model-agnostic meta-learning algorithms: Recurring and unseen tasks. In: NeurIPS (2021)
13. Feng, X., Zheng, Z., Mohapatra, P., Cansever, D.: A stackelberg game and markov modeling of moving target defense. In: International Conference on Decision and Game Theory for Security. pp. 315–335. Springer (2017)
14. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: International conference on machine learning (ICML). pp. 1126–1135 (2017)
15. Fujimoto, S., Hoof, H., Meger, D.: Addressing function approximation error in actor-critic methods. In: International conference on machine learning (ICML). pp. 1587–1596 (2018)
16. Ghavamzadeh, M., Mannor, S., Pineau, J., Tamar, A.: Bayesian reinforcement learning: A survey. Foundations and Trends in Machine Learning **8**(5-6), 359–492 (2015)
17. Hu, J., Wellman, M.P.: Nash q-learning for general-sum stochastic games. Journal of Machine Learning Research **4**, 1039–1069 (2003)
18. Huang, P., Xu, M., Fang, F., Zhao, D.: Robust reinforcement learning as a stackelberg game via adaptively-regularized adversarial training. arXiv preprint arXiv:2202.09514 (2022)
19. Jackson, T., Salamat, B., Homescu, A., Manivannan, K., Wagner, G., Gal, A., Brunthaler, S., Wimmer, C., Franz, M.: Compiler-generated software diversity. In: Moving Target Defense, pp. 77–98. Springer (2011)
20. Jafarian, J.H., Al-Shaer, E., Duan, Q.: Openflow random host mutation: transparent moving target defense using software defined networking. In: Proceedings of the first workshop on Hot topics in software defined networks (HotSDN). pp. 127–132 (2012)
21. Jajodia, S., Ghosh, A.K., Swarup, V., Wang, C., Wang, X.S.: Moving target defense: creating asymmetric uncertainty for cyber threats, vol. 54. Springer Science & Business Media (2011)
22. Könönen, V.: Asymmetric multiagent reinforcement learning. Web Intelligence and Agent Systems: An International Journal (WIAS) **2**(2), 105–121 (2004)
23. Li, H., Shen, W., Zheng, Z.: Spatial-temporal moving target defense: A markov stackelberg game model. In: International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS) (2020)
24. Li, H., Zheng, Z.: Optimal timing of moving target defense: A stackelberg game model. In: IEEE Military Communications Conference (MILCOM). IEEE (2019)
25. Luo, Y.B., Wang, B.S., Wang, X.F., Hu, X.F., Cai, G.L., Sun, H.: Rpah: Random port and address hopping for thwarting internal and external adversaries. In: 2015 IEEE Trustcom/BigDataSE/ISPA. vol. 1, pp. 263–270. IEEE (2015)
26. Mell, P., Scarfone, K., Romanosky, S.: Common vulnerability scoring system. IEEE Security & Privacy **4**(6), 85–89 (2006)
27. Nichol, A., Achiam, J., Schulman, J.: On first-order meta-learning algorithms. arXiv preprint arXiv:1803.02999 (2018)
28. Paruchuri, P., Pearce, J.P., Marecki, J., Tambe, M., Ordonez, F., Kraus, S.: Playing games for security: An efficient exact algorithm for solving bayesian stackelberg games. In: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems (AAMAS). pp. 895–902 (2008)
29. Paulin, A.: Secure sql server-enabling secure access to remote relational data. arXiv preprint arXiv:1201.1081 (2012)

30. Peng, W., Li, F., Huang, C.T., Zou, X.: A moving-target defense strategy for cloud-based services with heterogeneous and dynamic attack surfaces. In: International Conference on Communications (ICC). pp. 804–809. IEEE (2014)
31. Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., Dormann, N.: Stable-baselines3: Reliable reinforcement learning implementations. Journal of Machine Learning Research (2021)
32. Saputro, N., Tonyali, S., Aydeger, A., Akkaya, K., Rahman, M.A., Uluagac, S.: A review of moving target defense mechanisms for internet of things applications. Modeling and Design of Secure Internet of Things pp. 563–614 (2020)
33. Sengupta, S., Kambhampati, S.: Multi-agent reinforcement learning in bayesian stackelberg markov games for adaptive moving target defense. arXiv preprint arXiv:2007.10457 (2020)
34. Sengupta, S., Vadlamudi, S.G., Kambhampati, S., Doupé, A., Zhao, Z., Taguinod, M., Ahn, G.J.: A game theoretic approach to strategy generation for moving target defense in web applications. In: International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS). pp. 178–186 (2017)
35. Sharma, D.P., Kim, D.S., Yoon, S., Lim, H., Cho, J.H., Moore, T.J.: Frvm: Flexible random virtual ip multiplexing in software-defined networks. In: IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). pp. 579–587. IEEE (2018)
36. Sinha, A., Nguyen, T.H., Kar, D., Brown, M., Tambe, M., Jiang, A.X.: From physical security to cybersecurity. Journal of Cybersecurity **1**(1), 19–35 (2015)
37. von Stengel, B., Zamir, S.: Leadership with commitment to mixed strategies. CDAM Research Report LSE-CDAM-2004-01 (2004)
38. Taguinod, M., Doupé, A., Zhao, Z., Ahn, G.J.: Toward a moving target defense for web applications. In: 2015 IEEE International Conference on Information Reuse and Integration. pp. 510–517. IEEE (2015)
39. Thomas, S., Williams, L.: Using automated fix generation to secure sql statements. In: International Workshop on Software Engineering for Secure Systems (SESS). IEEE (2007)
40. Thompson, M., Evans, N., Kisekka, V.: Multiple os rotational environment an implemented moving target defense. In: The 7th International Symposium on Resilient Control Systems (ISRCS). pp. 1–6. IEEE (2014)
41. Vorobeychik, Y., Singh, S.: Computing stackelberg equilibria in discounted stochastic games (corrected version). In: Twenty-Sixth Conference on Artificial Intelligence (AAAI) (2012)
42. Vu, Q.L., Alumbaugh, Z., Ching, R., Ding, Q., Mahajan, A., Chasnov, B., Burden, S., Ratliff, L.J.: Stackelberg policy gradient: Evaluating the performance of leaders and followers. In: ICLR 2022 Workshop on Gamification and Multiagent Solutions (2022)
43. Wang, J.X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J.Z., Munos, R., Blundell, C., Kumaran, D., Botvinick, M.: Learning to reinforcement learn. arXiv preprint arXiv:1611.05763 (2016)
44. Weng, L.: Meta-learning: Learning to learn fast. lilianweng.github.io (2018), https://lilianweng.github.io/posts/2018-11-30-meta-learning/
45. Xie, Q., Chen, Y., Wang, Z., Yang, Z.: Learning zero-sum simultaneous-move markov games using function approximation and correlated equilibrium. In: COLT (2020)
46. Zhang, Y., Li, M., Bai, K., Yu, M., Zang, W.: Incentive compatible moving target defense against vm-colocation attacks in clouds. In: IFIP International Information Security Conference. pp. 388–399. Springer (2012)