# Optimal Attack and Defense on Reinforcement Learning

**Jeremy McMahan, Young Wu, Xiaojin Zhu, Qiaomin Xie**

University of Wisconsin-Madison
jmcmahan@wisc.edu, yw@cs.wisc.edu, jerryzhu@cs.wisc.edu, qiaomin.xie@wisc.edu

## Abstract

To ensure the usefulness of Reinforcement Learning (RL) in real systems, it is crucial to ensure they are robust to noise and adversarial attacks. In adversarial RL, an external attacker has the power to manipulate the victim agent's interaction with the environment. We study the full class of online manipulation attacks, which include (i) state attacks, (ii) observation attacks (which are a generalization of perceived-state attacks), (iii) action attacks, and (iv) reward attacks. We show the attacker's problem of designing a stealthy attack that maximizes its own expected reward, which often corresponds to minimizing the victim's value, is captured by a Markov Decision Process (MDP) that we call a meta-MDP since it is not the true environment but a higher level environment induced by the attacked interaction. We show that the attacker can derive optimal attacks by planning in polynomial time or learning with polynomial sample complexity using standard RL techniques. We argue that the optimal defense policy for the victim can be computed as the solution to a stochastic Stackelberg game, which can be further simplified into a partially-observable turn-based stochastic game (POTBSG). Neither the attacker nor the victim would benefit from deviating from their respective optimal policies, thus such solutions are truly robust. Although the defense problem is NP-hard, we show that optimal Markovian defenses can be computed (learned) in polynomial time (sample complexity) in many scenarios.

## 1 Introduction

Reinforcement Learning (RL) has become a staple with a plethora of applications including the breakthrough ChatGPT (Ouyang et al. 2022). With the growth of RL applications, it is critical to understand the security threats posed to RL and how to defend against them. In many applications, noisy measurements can cause the agent-environment interaction to evolve entirely differently than what one would expect in theory. Even worse, malicious attackers can strategically modify the agent-environment interaction to induce catastrophic outcomes for the agent. If RL methods are to be used in diverse and critical settings, it is essential to ensure these RL algorithms are robust to potential attacks.

In adversarial RL, a victim agent interacts with an environment while being disrupted by an attacker. The attacker has the power to manipulate each aspect of the victim-environment interaction. In particular, the attacker can change: (i) the environment's state (*state attacks*), (ii) the victim's observation (*observation attacks*), (iii) the action taken by the victim (*action attacks*), and (iv) the reward received by the victim (*reward attacks*). When the environment is fully-observable, observation attacks translate to well-studied *perceived-state attacks*. We refer to all of these attack surfaces by *online manipulation attacks*. The attacker may use a subset or all of these attack surfaces to optimize its own expected reward from the attack, which often corresponds to minimizing the victim's value. However, the attacker cannot perform arbitrary manipulations without raising suspicion. Hence, the attacker must restrict its manipulations to a predefined set of stealthy attacks. On the other hand, the attacker-aware victim seeks to choose a *defense* policy whose value is provably robust even under the worst possible stealthy attack.

From the attacker's perspective, it faces an optimal control problem: it needs to strategically choose stealthy attacks to optimize its value. Unlike typical control problems, the attacker must deal with the uncertainty of the victim's actions in addition to that of the stochastic environment. Thus, the attacker's problem involves a multi-agent feature. For any fixed victim policy $\pi$, we can view the attacker's problem as computing its best response attack to the victim's chosen $\pi$. From the victim's perspective, we argue it faces a Stochastic Stackelberg game: it needs to choose a policy that achieves maximum value in the environment under the attacker's best-response attack. A defense policy designed following this principle ensures neither the victim nor the attacker would benefit from deviating from their chosen policies, and so an equilibrium would be achieved. This implies that regardless of the attack, the defense policy always achieves at least the game's optimal value. However, computing optimal Stackelberg strategies for stochastic games is NP-hard. Thus, both the attacker and the victim are faced with challenging optimization problems.

Although the attack and defense problems are of great importance, complete solutions have yet to be discovered. For the attack problem, most works focus on the empirical aspects, lacking theoretical guarantees. Provably optimal attacks have only been devised for the special case of test-time, perceived-state attacks (Russo and Proutiere 2021;

Zhang et al. 2020a; Sun et al. 2022). The situation is even worse for the defense problem, which is arguably more important. Nearly all proposed defenses are designed to be effective against a specific, known attack. This results in a cat-and-mouse game: the attacker can just design a new attack for the given defense policy and so the victim would always be at risk. In addition, the two approaches with provable guarantees are restricted to the planning, reward-poisoning setting (Banihashem, Singla, and Radanovic 2021), and the test-time, perceived-state attack setting (Zhang et al. 2021). Furthermore, neither defense can be computed efficiently and it is unrealistic to assume the victim knows the attacker's exact algorithm.

**Our Contributions.** Despite the challenges of the attack and defense problems, we develop frameworks for computing optimal attacks and defenses for any combination of attack surfaces, which are provably efficient in many cases. From the attacker's side, we show that for any fixed victim policy, the optimal attack can be computed as the solution to another Markov Decision Process (MDP). We call this environment a *meta-MDP* since it is not the true environment, but is a higher-level environment induced by the victim-attacker-environment interaction. Importantly, the attacker can simulate an interaction with the meta-MDP by interacting with the victim and the true environment. Hence, the attacker can attack optimally by solving the meta-MDP using any standard MDP planning or RL algorithms. In addition, we show that the size of the meta-MDP is polynomial in the size of the original environment and the size of the victim's policy. Thus, optimal attacks can always be computed or learned efficiently. Our framework also extends to linear MDPs. Hence, we provide the first provably optimal attacks for beyond perceived-state attacks and the first provably optimal attacks for the linear setting, all of which can be computed in polynomial time. We note our framework also solves the certifying robustness problem posed in (Wu et al. 2022).

On the victim's side, we argue that the defense problem is most naturally modeled by a stochastic Stackelberg game (Vorobeychik and Singh 2021), which can be captured by a much simpler partially-observable turn-based stochastic game (POTBSG) (Hansen, Miltersen, and Zwick 2013). Thus, the victim can compute its optimal robust defense by finding a weak Stackelberg equilibrium (WSE) for the meta-POTBSG. Again, the victim can simulate the meta-POTBSG by interacting with the attacker and the true environment. When the attacker is adversarial, the victim can defend optimally by solving the meta-POTBSG using any standard zero-sum POTBSG planning or distributed learning algorithms. Unlike the attack problem, we show that the victim's defense problem is NP-hard in general even to find approximate solutions when observation attacks are permitted. However, we show that optimal Markovian defenses can be computed efficiently when excluding observation attacks by exploiting the sequential nature of the attacks. This gives a broad class of games for which WSE is computable. Overall, we present the first-ever provable defense algorithms for both the planning and learning settings and show our defenses can be computed efficiently for a broad class of instances.

## Related Work

Many prior works have studied adversarial RL under various models and objectives. Amongst the first works, Behzadan and Munir (2017); Huang et al. (2017); Kos and Song (2017) study perceived-state attacks through the lens of adversarial examples for deep neural nets (Goodfellow, Shlens, and Szegedy 2014). Kos and Song (2017) also considers adversarial examples, but with the goal of minimizing the number of attacks needed to achieve large damage. These works focused on achieving large damage at the current time. Later Lin et al. (2017); Sun et al. (2020) developed more advanced heuristics that incorporate future value into their attacks to achieve long-term damage. Meanwhile, Tretschk, Oh, and Fritz (2018) trained an adversarial deep net to compute perturbations that allows other objectives for the attacker.

Afterward, many works began considering the objective of maximizing the damage to the victim rather than minimizing the number of attacks. Russo and Proutiere (2021); Zhang et al. (2020a) developed optimal algorithms for computing perceived-state attacks. Both works formulated the attack problem as a different MDP as we do here. Sun et al. (2022) formulated an actor-director model for the attack problem that is easier to solve for some MDPs and retains guarantees of optimality. The idea of adversarial training was then used in conjunction with the attack formulation from (Zhang et al. 2020a) to obtain experimentally robust victim policies (Zhang et al. 2021).

Action and reward attacks have been considered heavily in the training-time setting. For example, Tessler, Efroni, and Mannor (2019); Lee et al. (2021) considered action attacks. Reward poisoning attacks are the focus of the work by Zhang et al. (2020b); Rangi et al. (2022). In fact, a combination action and reward attack are devised by Rangi et al. (2022). Most of these works consider the policy teaching setting, where the attacker's goal is for the victim to follow a fixed policy $\pi^\dagger$. Some algorithms achieve sublinear regret for the attacker when the victim policy is no regret (Liu and Lai 2021); though, none compute truly optimal attacks.

# 2 Attack Surfaces

**POMDPs.** We denote a infinite-horizon discounted environment POMDP by $M = (\mathcal{S}, \mathcal{O}, \mathcal{A}, P, R, \gamma, \mu)$ where (i) $\mathcal{S}$ is the state set, (ii) $\mathcal{O}$ is the observation set, (iii) $\mathcal{A}$ is the action set, (iv) $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition kernel, (v) $R : \mathcal{S} \times \mathcal{A} \to \Delta(\mathbb{R})$ is the reward distribution, (vi) $\gamma$ is the discount factor, and (vii) $\mu \in \Delta(S)$ is the initial state distribution. We let $\mathcal{O}(s)$ denote the distribution of observations at state $s$. We also let $\mathcal{R}$ denote the set of all supported rewards. The total expected reward the victim receives from following policy $\pi$ in environment $M$ is its *value*, i.e., the expected cumulative discounted rewards $V_M^\pi := \mathbb{E}_M^\pi \left[\sum_{t=0}^\infty \gamma^t r(s_t, a_t)\right]$.

Suppose the victim interacts with a Markovian environment, $M$, using a fixed stationary, Markovian policy $\pi : \mathcal{O} \to \Delta(A)$. At any time $t$, let $s_t$ denote $M$'s current state

and $o_t$ denote the generated observation. In the standard setting, the victim chooses an action $a_t \sim \pi(o_t)$ and then receives a reward $r_t \sim R(s_t, a_t)$. Afterwards, $M$ transitions to its next state $s_{t+1} \sim P(s_t, a_t)$. We see there are several points during time $t$ at which information is exchanged between the victim and $M$. We further break down the interaction at time $t$ based on these points of information exchange, which we call *subtimes*:

1. At the first subtime, $t_1$, $M$ receives its state $s_t \sim P(s_{t-1}, a_{t-1})$.
2. At the second subtime, $t_2$, the victim receives its observation $o_t \sim \mathcal{O}(s_t)$.
3. At the third subtime, $t_3$, $M$ receives the victim's action $a_t \sim \pi(o_t)$.
4. At the fourth subtime, $t_4$, the victim receives its reward $r_t \sim R(s_t, a_t)$.

**Online Attacks.** In the adversarial setting, a third-party called the *attacker* interferes with the victim-$M$ interaction. Here, the attacker may intercept and then corrupt the information being exchanged between the victim and environment $M$. The attacker has access to four attack surfaces:

1. (*State Attack*) A state attack changes the state of $M$ from $s_t$ to $s_t^\dagger$. The attack influences the observation $o_t \sim \mathcal{O}(s_t^\dagger)$. If $M$ receives action $a_t$, the attack also influences the reward $r_t \sim R(s_t^\dagger, a_t)$ and the next state $s_{t+1} \sim P(s_t^\dagger, a_t)$.
2. (*Observation Attack*) An observation attack causes the victim to receive observation $o_t^\dagger$ instead of $o_t \sim \mathcal{O}(s_t)$. The attack influences the victim's action $a_t \sim \pi(o_t^\dagger)$.
3. (*Action Attack*) An action attack causes $M$ to receive action $a_t^\dagger$ instead of $a_t$. The attack influences the reward $r_t \sim R(s_t, a_t^\dagger)$ and next state $s_{t+1} \sim P(s_t, a_t^\dagger)$.
4. (*Reward Attack*) A reward attack causes the victim to receive reward $r_t^\dagger$ instead of reward $r_t \sim R(s_t, a_t)$.

We call each of these attack surfaces as *online manipulation attacks*. These attack surfaces in conjunction give the attacker the power to corrupt every element of the triple $(s, a, r)$ that define the interaction between the victim and $M$.

If $M$ is fully observable, observation attacks correspond to *perceived-state* attacks, which change what the victim thinks is $M$'s state. Notice unlike the other surfaces, state attacks could be performed at two different subtimes. Namely, the attacker can change the state before $M$ transitions at $t_1$ or before $M$ receives the victim's action at $t_3$. For simplicity, we assume state attacks only happen at $t_1$, but our results apply equally well to both versions.

**Adversarial Interaction.** Overall, the victim-attacker-$M$ interaction at time $t$ now evolves as follows:

1. At subtime $t_1$, $M$ is in state $s_t$.
   (a) Attacker: changes $s_t$ to $s_t^\dagger$.
   (b) $M$: enters state $s_t^\dagger$ and generates observation $o_t \sim \mathcal{O}(s_t^\dagger)$.
2. At subtime $t_2$, $M$ is in state $s_t^\dagger$ and has generated observation $o_t$.
   (a) Attacker: changes $o_t$ to $o_t^\dagger$.
   (b) Victim: chooses action $a_t \sim \pi(o_t^\dagger)$.
3. At subtime $t_3$, $M$ is in state $s_t^\dagger$ and the victim chose action $a_t$.
   (a) Attacker: changes $a_t$ to $a_t^\dagger$.
   (b) $M$: generates reward $r_t \sim R(s_t^\dagger, a_t^\dagger)$ and generates state $s_{t+1} \sim P(s_t^\dagger, a_t^\dagger)$.
4. At subtime $t_4$, $M$ has generated reward $r_t$.
   (a) Attacker: changes $r_t$ to $r_t^\dagger$.
   (b) Victim: receives reward $r_t^\dagger$.

This process then repeats starting from $s_{t+1}$.

**Attacker Constraints.** In general, the attacker may not arbitrarily manipulate the interaction. For example, some attacks may be physically impossible or risk detection. As such, we assume the attacker has a set $\mathcal{B}$ that defines the feasible manipulations it can perform. For example, the attacker might require a manipulated observation to be visually similar to the true observation. Thus, the set of feasible observation attacks should depend on the true observation. Applying the same logic to each attack surface, we see the feasible attack sets should take the form: $\mathcal{B}(s) \subseteq \mathcal{S}$, $\mathcal{B}(o) \subseteq \mathcal{O}$, $\mathcal{B}(a) \subseteq \mathcal{A}$, and $\mathcal{B}(r) \subseteq \mathcal{R}$. However, in some cases, the feasibility of an attack would depend on the interaction before the attack, not just the current element being manipulated. To be fully general, we allow the feasibility sets to take the form: at subtime $t_1$, $\mathcal{B}(s) \subseteq \mathcal{S}$; at subtime $t_2$, $\mathcal{B}(s, o) \subseteq \mathcal{O}$; at subtime $t_3$, $\mathcal{B}(s, o, a) \subseteq \mathcal{A}$; and, at subtime $t_4$, $\mathcal{B}(s, o, a, r) \subseteq \mathcal{R}$.

## 3 Optimal Attacks

**Attacker's Goal.** In Section 2, we saw how an attacker can disrupt an interaction but haven't discussed why it would do this. Suppose the attacker has a reward function $g(s, a, r)$ that depends on the victim's received reward, possibly in addition to $M$'s state and the victim's action. The attacker's goal is then to construct an attack that maximizes its own expected reward. Commonly, an attacker just wants to minimize the victim's expected reward under attack, or equivalently maximize the damage to the victim's expected reward. In this case, the attacker's reward function is $g(s, a, r) = -r$. Alternatively, the attacker may want the victim to behave in a specified way. This goal is equivalent to the attacker wanting the victim to choose actions that match a fixed target policy $\pi^\dagger$ as often as possible. In this case, the attacker's reward function is $g(s, a, r) = \mathbf{1}\left\{a = \pi^\dagger(s)\right\}$.

**Definition 1** (Attack Problem)**.** For any $\pi$, the attacker's seeks a policy $\nu^* \in N$ that maximizes its expected reward from the victim-attacker-$M$ interaction:

$$\nu^* \in \arg\max_{\nu \in N} \mathbb{E}_M^{\pi,\nu}\left[\sum_{t=0}^{\infty} \gamma^t g(s_t, a_t, r_t)\right]. \qquad (1)$$

We show that the attacker's problem is captured by a MDP. The key insight is that by defining the attacker's state set to capture the results of previous attacks from $t_1$ up to the current subtime, then each attack becomes Markovian with respect to the expanded state set. This is not a significant burden on the attacker since it would need to keep track of this information anyway to compute the feasible attack sets. Thus, the attacker just needs to keep track of the information within a time step to compute optimal attacks.

**Definition 2** (Meta-MDP)**.** For any victim policy $\pi$, the attacker's meta-MDP is $\overline{M} = (\overline{\mathcal{S}}, \overline{\mathcal{A}}, \overline{P}, \overline{r}, \overline{\gamma}, \overline{\mu})$ where,

- $\overline{\mathcal{S}} = \mathcal{S} \cup (\mathcal{S} \times \mathcal{O}) \cup (\mathcal{S} \times \mathcal{O} \times \mathcal{A}) \cup (\mathcal{S} \times \mathcal{O} \times \mathcal{A} \times \mathcal{R})$.
- $\overline{\mathcal{A}}(s) = \mathcal{B}(s)$, $\overline{\mathcal{A}}(s,o) = \mathcal{B}(s,o)$, $\overline{\mathcal{A}}(s,o,a) = \mathcal{B}(s,o,a)$, and $\overline{\mathcal{A}}(s,o,a,r) = \mathcal{B}(s,o,a,r)$.
- The transitions vary per subtime. Let $\overline{s} \in \overline{\mathcal{S}}$, $\overline{a} \in \overline{\mathcal{A}}(\overline{s})$, and $\overline{s}' \in \overline{\mathcal{S}}$.
  1. If $\overline{s} = s$, then $\overline{a} = s^\dagger$ and $\overline{s}' = (s^\dagger, o)$:
  $$\overline{P}(\overline{s}' \mid \overline{s}, \overline{a}) = \mathcal{O}(o \mid s^\dagger).$$
  2. If $\overline{s} = (s,o)$, then $\overline{a} = o^\dagger$ and $\overline{s}' = (s, o^\dagger, a)$:
  $$\overline{P}(\overline{s}' \mid \overline{s}, \overline{a}) = \pi(a \mid o^\dagger).$$
  3. If $\overline{s} = (s,o,a)$, then $\overline{a} = a^\dagger$ and $\overline{s}' = (s, o, a^\dagger, r)$:
  $$\overline{P}(\overline{s}' \mid \overline{s}, \overline{a}) = R(r \mid s, a^\dagger).$$
  4. If $\overline{s} = (s,o,a,r)$, then $\overline{a} = r^\dagger$ and $\overline{s}' = s'$:
  $$\overline{P}(\overline{s}' \mid \overline{s}, \overline{a}) = P(s' \mid s, a).$$

  All other transitions have probability 0.
- Let $\overline{s} \in \overline{\mathcal{S}}$, and $\overline{a} \in \overline{\mathcal{A}}(\overline{s})$. If $\overline{s} = (s,o,a,r)$ and $\overline{a} = r^\dagger$, then $\overline{r}(\overline{s}, \overline{a}) = g(s, a, r^\dagger)$. For all other meta-states, $\overline{r}(\overline{s}, \overline{a}) = 0$.
- $\overline{\gamma} = \gamma^{1/4}$.
- $\overline{\mu}(s) = \mu(s)$ for $s \in \mathcal{S}$ and $\overline{\mu}(\overline{s}) = 0$ otherwise.

**Reward Subtlety.** Note that the attacker only receives a reward at every fourth subtime. This means the discount factor has to be "slowed down" so that the factor at every fourth time step matches that of each single time step of $M$. Specifically, choosing $\overline{\gamma} = \gamma^{1/4}$ ensures that $\overline{\gamma}^{4t} = \gamma^t$.

**Proposition 1.** *The maximum expected reward the attacker can achieve from any attack on $\pi$ is $V^*_{\overline{M}}$, the maximum expected total discounted reward for the meta-MDP $\overline{M}$. Furthermore, any optimal deterministic, stationary policy $\nu^*$ for $\overline{M}$ is an optimal attack policy.*

**Online Interaction.** Suppose the attacker has computed some attack policy $\nu$ from $\overline{M}$. In order to use $\nu$ to interact with the victim and $M$, the attacker must know the meta-state at any given subtime. As long as the attacker can observe the interaction between the victim policy $\pi$ and $M$, it can effectively simulate the interaction with the meta-MDP $\overline{M}$ online using a constant amount of memory. At time $t$, the attacker only needs to store $s_t$, $o_t$, $a_t$, and $r_t$ when they are revealed to the attacker. With this information, the attacker knows the meta-state for each subtime and so can apply $\nu$ to determine its next attack. Upon reach the next time $t+1$, the attacker can forget $s_t$, $o_t$, $a_t$, and $r_t$ and start from $s_{t+1}$. See Algorithm 1.

Algorithm 1: Attacker Interaction Protocol

**Input:** $(\pi, \nu)$

**for** $t = 1 \ldots$ **do**
- Attacker sees $s_t$, and computes a state attack $s_t^\dagger = \nu(s_t)$
- Attacker sees $o_t \sim \mathcal{O}(s_t^\dagger)$, and computes an observation attack $o_t^\dagger = \nu(s_t, o_t)$
- Attacker sees $a_t \sim \pi(o_t^\dagger)$, and computes an action attack $a_t^\dagger = \nu(s_t, o_t, a_t)$
- Attacker sees $r_t \sim \mathcal{R}(s_t^\dagger, a_t^\dagger)$, and computes a reward attack $r_t^\dagger = \nu(s_t, o_t, a_t, r_t)$
- Attacker receives reward $g(s_t^\dagger, a_t^\dagger, r_t^\dagger)$, and forgets $(s_t, o_t, a_t, r_t)$

**end for**

**Solving $\overline{M}$.** If the attacker has full knowledge of $M$ and the victim's policy $\pi$, then the attacker has all the knowledge needed to construct the meta-MDP $\overline{M}$. Once $\overline{M}$ is constructed, the attacker can use any planning algorithm, such as policy iteration, to compute the optimal attack. Alternatively, if the attacker does not know $M$ and $\pi$, it can still simulate interacting with $\overline{M}$ online as described before to perform learning. In particular, the attacker can replace the call to $\nu$ in Algorithm 1 with any off-the-shelf learning algorithm. For the episodic setting, we view the attacker as attacking a new victim following the same policy $\pi$ in each episode.

Observe that $|\overline{\mathcal{S}}| \leq |\mathcal{S}||\mathcal{O}||\mathcal{A}||\mathcal{R}|$, $|\overline{\mathcal{A}}| \leq |\mathcal{S}|+|\mathcal{O}|+|\mathcal{A}|+|\mathcal{R}|$, and $\overline{\gamma} = \gamma^{1/4}$. Thus, whenever $M$'s rewards are finitely supported, $|\overline{M}| = poly(|M|)$, where $|M|$ is the total size of $M$'s description. As such, any polytime planning algorithm or polynomial sample-complexity learning algorithm applied to $\overline{M}$ yields an algorithm for computing optimal attacks that has polynomial complexity.

**Proposition 2.** *When $M$'s rewards have finite support or no reward attacks are allowed, $|\overline{M}| = poly(|M|)$. Thus, an optimal attack policy can be computed in polynomial time by planning in $\overline{M}$, and learning an optimal attack policy can be performed with polynomial sample complexity by learning in $\overline{M}$.*

**Remark 1** (Restricted Surfaces)**.** By restricting $\overline{\mathcal{A}}$ to singleton sets (e.g. set $\overline{\mathcal{A}}(s,o,a) = \{a\}$ to disallow action attacks), $\overline{M}$ recovers optimal attacks for each individual surface as well as attacks for any subset of available attack surfaces. This captures all standard test-time attacks, generalizing the perceived-state attack MDP of (Zhang et al. 2020a). We also note if the attacker does not perform reward attacks, $\overline{M}$ can be modified to avoid $\mathcal{R}$ and so $M$ having finite supported rewards is unnecessary in the complexity results.

One might ask whether the perceived-state attack MDP defined in (Zhang et al. 2020a) would work in the linear setting. We point out that the transition takes the following

form,

$$\tilde{P}(s' \mid s, s^\dagger) = \mathbb{E}_{a\sim\pi(s^\dagger)} P(s' \mid s, a)$$
$$= \int_a P(s' \mid s, a)\pi(a \mid s^\dagger)da.$$

As $\pi$ and $P$ are multiplied together, $\tilde{P}$ would be a quadratic transition. On the other hand, our particular choice of subtimes induces linear structure in $\overline{M}$. Specifically, each transition of $\overline{P}$ is defined by a single distribution involving $\pi$ or $M$. If both $\pi$ and $M$ have a linear structure, then so will $\overline{M}$. Then, $\overline{M}$ can be solved by standard linear RL algorithms. Thus, so long as $\pi$ is linear, the attacker can compute optimal attacks on linear environments.

**Theorem 1.** *If $M$ is linear and $\pi$ is linear, then $\overline{M}$ is linear. Furthermore, the dimension of $\overline{M}$, $d(\overline{M})$, is at most $\max\{d(\pi), d(M)\} + 1$. Thus, if $\pi$ is linear, optimal attacks on linear environments can be computed or learned efficiently*

**Remark 2** (Beyond Markovian Policies)**.** Our construction can be easily modified to handle non-Markovian victim policies. If the victim uses some finite amount of past history $\tilde{\mathcal{H}}$, we simply modify the meta-state space to remember the same amount of past history and adjust the construction appropriately. The size of $\overline{M}$ is now a polynomial in both $|M|$ and the size of the policy when described explicitly as a mapping from histories to action distributions. We defer the details to the Appendix.

## 4 Optimal Defense

Now that we have seen how the attacker can best attack, it begs the question of how the victim should defend against attacks. Intuitively, the victim should choose a defense policy that is robust to attack. However, it does not suffice to just be robust against a particular attack. In fact, the attacker could lie about its attack algorithm to bait the victim into choosing a policy that actually benefits the attacker. Even if some attacker does use that particular attack algorithm, other attackers may employ different methods that lead the victim to poor value. As new attacks are formulated, the victim would have to constantly create more complex policies designed with all known attacks in mind. This would become a never-ending cat-and-mouse game during which the victim's policy will often be at risk of new attacks. Thus, for a policy to be satisfactorily robust, we require it to be robust against the worst possible attack. This way, no matter what future strategies an attacker may use, the victim is already prepared.

We can formalize this intuition using the Stackelberg approach for Security Applications (Korzhyk, Conitzer, and Parr 2010). For any $\pi$ and $\nu$, let $V_1^{\pi,\nu}$ and $V_2^{\pi,\nu}$ denote the victim's and attacker's expected reward respectively under the victim-attacker-$M$ interaction induced by $\pi$ and $\nu$. Note, both of these quantities can be computed efficiently using the techniques from Section 3. Let $V_1$ and $V_2$ denote infinite matrices whose $(\pi, \nu)$ entry corresponds to $V_1^{\pi,\nu}$ and $V_2^{\pi,\nu}$ respectfully. We define an infinite bimatrix game $G$ whose payoff matrices are $(V_1, V_2)$. For any fixed victim $\pi$, it is clear that a rational attacker would play some best-response policy, $\nu \in BR(\pi) := \max_{\nu\in N} V_2^{\pi,\nu}$. Thus, an optimal defense policy is exactly an optimal Stackelberg strategy for player 1 in $G$ (Conitzer 2015).

**Definition 3** (Defense Problem)**.** The victim seeks a policy $\pi^*$ that maximizes its expected reward from the victim-attacker-$M$ interaction under the worst-case attack:

$$\pi^* \in \max_{\pi\in\Pi} \min_{\nu\in BR(\pi)} V_1^{\pi,\nu}. \tag{2}$$

Observe that this solution is truly robust: by definition, the attacker given $\pi$ would never want to deviate from $BR(\pi)$, and similarly, by definition the victim would never want to deviate from its defense policy when assuming the worst possible attack. Thus, we consider such attack and defense policies as truly *optimal*. However, as the victim faces partial observability, an optimal defense for the victim is history-dependent in general. Consequently, the attacker's best response must also be history-dependent. Thus, $\Pi$ and $N$ consist of history-dependent policies in the definition above.

Although optimal Stackelberg strategies for Stochastic games are generally difficult to compute (Letchford et al. 2021), we can exploit the special structure of the victim-attacker-$M$ interaction to develop useful algorithms. Recall that at subtime $t_2$ in Algorithm 1, the attacker changes the observation to $o^\dagger$, and then the victim chooses an action $a = \pi(o^\dagger)$. If we simply give the victim the autonomy to choose any action $a$ at this point rather than according to a fixed policy $\pi$, then this interaction evolves like a turn-based game. In fact, we show this game can be modeled as a partially observable turn-based stochastic game (POTBSG) (Zheng, Jung, and Lin 2022). POTBSGs exhibit much more structure than a general imperfect-information stochastic game, so enable more efficient solution methods. We see the construction is almost identical to Definition 2.

**Definition 4.** The victim-attacker's POTBSG is $\overline{G} = (\overline{\mathcal{S}}_1 \cup \overline{\mathcal{S}}_2, \overline{\mathcal{O}}, \overline{\mathcal{A}}, \overline{P}, \overline{r}, \overline{\gamma}, \overline{\mu})$ where,

- $\overline{\mathcal{S}}_1 := \mathcal{S} \times \mathcal{O} \times \{\varnothing\}$ and $\overline{\mathcal{S}}_2 := \mathcal{S} \cup (\mathcal{S} \times \mathcal{O}) \cup (\mathcal{S} \times \mathcal{O} \times \mathcal{A}) \cup (\mathcal{S} \times \mathcal{O} \times \mathcal{A} \times \mathcal{R})$.
- $\overline{\mathcal{O}}(\overline{s}) := o$ for $\overline{s} = (s, o, \varnothing)$ and $\overline{\mathcal{O}}(\overline{s}) := \overline{s}$ otherwise.
- $\overline{\mathcal{A}}(s) := \mathcal{B}(s)$, $\overline{\mathcal{A}}(s, o) := \mathcal{B}(s, o)$, $\overline{\mathcal{A}}(s, o, \varnothing) := \mathcal{A}$, $\overline{\mathcal{A}}(s, o, a) := \mathcal{B}(s, o, a)$, and $\overline{\mathcal{A}}(s, o, a, r) := \mathcal{B}(s, o, a, r)$.
- Let $\overline{s} \in \overline{\mathcal{S}}$, $\overline{a} \in \overline{\mathcal{A}}(\overline{s})$, and $\overline{s}' \in \overline{\mathcal{S}}$.
  1. If $\overline{s} = s$, then $\overline{a} = s^\dagger$ and $\overline{s}' = (s^\dagger, o)$: $\overline{P}(\overline{s}' \mid \overline{s}, \overline{a}) := \mathcal{O}(o \mid s^\dagger)$.
  2. If $\overline{s} = (s, o)$, then $\overline{a} = o^\dagger$ and $\overline{s}' = (s, o^\dagger, \varnothing)$: $\overline{P}(\overline{s}' \mid \overline{s}, \overline{a}) := \pi(a \mid o^\dagger)$.
  3. If $\overline{s} = (s, o, \varnothing)$, then $\overline{a} = a$ and $\overline{s}' = (s, o^\dagger, a)$: $\overline{P}(\overline{s}' \mid \overline{s}, \overline{a}) := 1$.
  4. If $\overline{s} = (s, o, a)$, then $\overline{a} = a^\dagger$ and $\overline{s}' = (s, o, a^\dagger, r)$: $\overline{P}(\overline{s}' \mid \overline{s}, \overline{a}) := R(r \mid s, a^\dagger)$.
  5. If $\overline{s} = (s, o, a, r)$, then $\overline{a} = r^\dagger$ and $\overline{s}' = s'$: $\overline{P}(\overline{s}' \mid \overline{s}, \overline{a}) := P(s' \mid s, a)$.

  All other transitions have probability 0.

- Let $\overline{s} \in \overline{\mathcal{S}}$, and $\overline{a} \in \overline{\mathcal{A}}(\overline{s})$. $\overline{r}_1(\overline{s}, \overline{a}) := r^\dagger$ and $\overline{r}_2(\overline{s}, \overline{a}) := g(s, a, r^\dagger)$ if $\overline{s} = (s, o, a, r)$ and $\overline{r}_1(\overline{s}, \overline{a}) := \overline{r}_2(\overline{s}, \overline{a}) := 0$ otherwise.
- $\overline{\gamma} := \gamma^{1/5}$.
- $\overline{\mu}(s) := \mu(s)$ for $s \in \mathcal{S}$ and $\overline{\mu}(\overline{s}) := 0$ otherwise.

Note that $\overline{\mathcal{S}}_1$ is the set of states in which the victim takes an action, and $\overline{\mathcal{S}}_2$ is the set of states in which the attacker takes an action. The observation and action set $\overline{\mathcal{O}}$ and $\overline{\mathcal{A}}$ as functions of the states are combined for the two players, and this implies that the observations and actions for the victim are $\overline{\mathcal{A}}(\overline{\mathcal{S}}_1)$ and $\overline{\mathcal{O}}(\overline{\mathcal{S}}_1)$, and for the attacker are $\overline{\mathcal{A}}(\overline{\mathcal{S}}_2)$ and $\overline{\mathcal{O}}(\overline{\mathcal{S}}_2)$. Observe that $V^{\pi,\nu}_{\overline{G},1} = V^{\pi,\nu}_1$ and $V^{\pi,\nu}_{\overline{G},2} = V^{\pi,\nu}_{\overline{M}}$ and so $G$ is just the normal-form representation of the POTBSG $\overline{G}$.

**Proposition 3.** *Any WSE for $\overline{G}$ yields an optimal defense policy.*

In general, methods to compute WSE are unknown. However, we show many settings where a WSE for $\overline{G}$ can be computed, even efficiently. First, suppose the attacker is completely adversarial so that $\overline{G}$ becomes a zero-sum game. In this case, it is known that $WSE = SSE = NE$. Thus, it suffices to compute an NE for a zero-sum POTBSG.

**Proposition 4.** *If the attacker is completely adversarial, an optimal defense policy can be computed as an NE of $\overline{G}$ using any planning or distributed learning algorithms for zero-sum POTBSGs.*

Note, it is important that the victim uses a distributed learning algorithm since it would not be able to see the attacker's manipulations, only the effects of the manipulations, nor be able to collaborate with the attacker. From Proposition 3, we see that the victim can compute an optimal defense policy to an adversarial attacker by computing any CCE to $\overline{G}$. However, even computing an approximately optimal Markovian policy against a fixed attack is equivalent to solving a POMDP, which is NP-hard (Lusena, Goldsmith, and Mundhenk 2001). Thus, computing near-optimal defenses is intractable in the worst case.

**Proposition 5.** *For any $\epsilon > 0$ an $\epsilon$-approximate optimal defense policy is NP-hard to compute even when restricting $\Pi$ and $N$ to be the class of Markovian policies.*

**Efficient Methods.** The main bottleneck to computing defenses efficiently in fully-observable systems is the presence of perceived-state attacks. Absent these attacks, the POTBSG specializes to a traditional TBSG, which is a special case of a stochastic game.

**Observation 1.** When $M$ is fully observable and the attacker cannot perform perceived-state attacks, $\overline{G}$ simplifies to a TBSG.

In the adversarial case, we see that $\overline{G}$ is simply a zero-sum TBSG. In zero-sum TBSGs, even stationary NE can be computed or learned efficiently (Cui and Yang 2021) unlike the case with CCE for MGs (Daskalakis, Golowich, and Zhang 2022) and the solutions are exact.

**Proposition 6.** *If $M$ is fully-observable, no perceived-state attacks are allowed, and $M$'s rewards have finite support (or no reward attacks are allowed), and the attacker is adversarial, then an* optimal stationary *defense policy can be computed in polynomial time and learned with polynomial sample complexity.*

Although it is unclear whether Markovian policies guarantee the victim as much value as history-dependent ones, Markovian policies are commonplace since they are easier to store and deploy in practice. In fact, for the finite-horizon planning setting, the attacker need not be restricted. We give polynomial time planning algorithms to compute an optimal defense so long as perceived-state attacks are banned. To our knowledge, this is the first non-trivial setting for which WSE can be computed efficiently and the first non-trivial setting for which SSE can be computed beyond single-period games.

**Theorem 2.** *If $M$ is fully-observable and has a finite horizon, no perceived-state attacks are allowed, and $M$'s rewards have finite support (or no reward attacks are allowed), then an optimal defense policy can be computed in polynomial time.*

The intuition is the victim can simulate the attacker's best-response function using backward induction. Once it knows the best response for a particular stage game, it can then brute-force find the best action to take at that stage. The key insight is that the attacker's best response is always deterministic since it gets to see the victim's realized actions. Thus, the victim also has no benefit from randomization. As such, the victim can brute-force compute its optimal deterministic action to take during a single stage and then propagate that solution backward to be used in previous times.

To illustrate this, we derive a backward induction algorithm for efficient defense against action attacks and present the full defense algorithm in the Appendix. Suppose the victim has already committed to $\{\pi^*_t\}_{t=h+1}^H$, where $H$ is the finite time-horizon. Clearly, for any choice of victim's action $a$, the attacker's best response to $a$ and the future partial policy is:

$$BR_h(s,a) = \arg\max_{a^\dagger \in \overline{\mathcal{A}}(s,a)} g_h(s, a, r_h(s,a)) + \mathbb{E}_{s' \sim P_h(s,a^\dagger)} V^*_{h+1,2}(s', \pi^*_{h+1}(s')),$$

where $V^*_{h,2}(s,a)$ is the maximum value achieved. Then, the victim can compute its best action for the stage game $(h,s)$ as a maximizer of,

$$V^*_{h,1}(s) = \max_{a \in \mathcal{A}} \min_{a^\dagger \in BR_h(s,a)} r_h(s, a^\dagger) + \mathbb{E}_{s' \sim P_h(s,a^\dagger)} V^*_{h+1,1}(s').$$

When the game is zero-sum, the algorithm is even simpler: the victim need not even simulate the attacker's best response. The recursion is simply:

$$V^*_{h,1}(s) = \max_{a \in \mathcal{A}} \min_{a^\dagger \in \overline{\mathcal{A}}(s,a)} r_h(s, a^\dagger) + \mathbb{E}_{s' \sim P_h(s,a^\dagger)} V^*_{h+1,1}(s').$$

The construction for defending against all non-perceived state surfaces is a bit more complicated but retains this same structure.

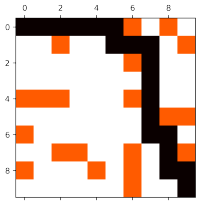

Figure 1: Optimal Policy Path.

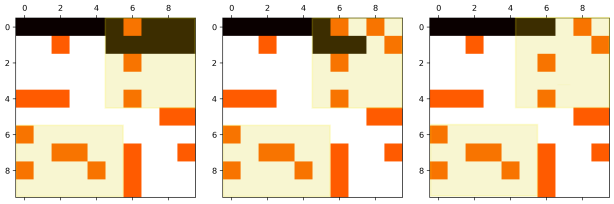

Figure 2: Attacked Paths.

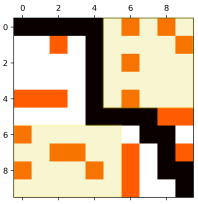

Figure 3: Defense Policy Path

**Remark 3** (Multi-Agent Extension)**.** We note that all of our results remain the same when multiple victims are present. This can be done without changing any of the previous notations by interpreting $\mathcal{A} = A_1 \times \ldots \times A_n$ as the joint action space and $\pi$ as a joint policy. From the attacker's perspective, attacking many victims just looks like attacking a single victim with a large action space. A WSE in $\overline{G}$ still breaks up into an independent joint policy for the victims and the attacker, but the joint policy may require the victims to correlate with each other.

## 5 Experiments

We illustrate our frameworks with a classical grid-world shortest path problem with obstacles. Here, each state is a cell in a $n \times n$ grid. Some grid cells are filled with lava and so dangerous to the victim. From any cell, the victim can move left (L), right (R), up (U), or down (D) so long as it remains on the grid. In addition, the victim can stay (S) in its current cell. The agent wishes to get from the top-left cell $(0, 0)$ to the bottom-right, "goal", cell $(n-1, n-1)$ as quickly as possible while avoiding lava. To capture this goal, we assume the victim receives a reward of 1 for entering the goal cell and continues to receive a reward of 1 for each time it remains there to incentivize the victim to reach the goal quickly. We also assume the victim receives a penalty reward of $-H$ whenever it enters a lava cell, where $H$ is the finite horizon.

Here, we test our methods on a $10 \times 10$ grid world with $H = 20$ so that the victim has enough time to reach the goal and stay there. We computed an optimal policy $\pi^*$ for the grid, which achieves the victim a value of 3. In Figure 1 we visualize $\pi^*$ through the path the victim follows when using $\pi^*$. The black cells represent a cell the victim entered during its interaction. The orange cells represent lava.

### Grid Attacks

The attacker can utilize its surfaces to disrupt the victim's path. For simplicity, assume that the attacker is purely adversarial and so it seeks to prevent the victim from reaching the goal and even trick it into lava cells if possible. Suppose that most of the grid is under security and so attacks cannot be safely made. The attacker is restricted to only attacking edges of the grid, which are not monitored. Here, the regions include the top-right subgrid and the bottom-left subgrid shaded in yellow. However, in those regions, it may use any attack it likes from its given surface.

In Figure 2, we see from left to right the path under an optimal perceived-state attack, true-state attack, and action attack. The agent receives $-100$, $0$, and $-160$ value from each attack respectively. In all cases, the victim no longer reaches the goal after getting attacked in the top-right subgrid. We see the perceived-state attack functions by tricking the agent into entering lava; whereas the action attack simply forces the victim into lava. On the other hand, the state attacks can transport the victim into lava, but they immediately leave and so suffers less damage than in the other attacks despite seeming to be the most powerful.

### Grid Defense

We see that if the victim simply follows $\pi^*$, the effects of attacks can be catastrophic. The victim knows the upper-right and bottom-left subgrids are not monitored and so can assume attacks are conducted there. Using this information, the defense algorithm yields a policy $\hat{\pi}$ that completely avoids the unsafe region. The victim still achieves the optimal value of 3 even under the strongest-possible attack. The new path under attack is illustrated in Figure 3. We see the robust path simply squeezes between the two unsafe regions.

## 6 Conclusions

In this paper, we rigorously studied the attack and defense problems of reinforcement learning. We showed that for any attack's surface, a malicious attacker can optimally and efficiently maximize its own rewards by solving a higher lever meta-MDP. Then, we formally defined the defense problem and showed it is a WSE of a POTBSG. In the zero-sum setting, we showed standard zero-sum MARL can be used to find optimal defense policies. When perceived-state attacks are not allowed, the victim can also compute an optimal defense policy in polynomial time using a robust backward induction algorithm. Although we present an optimal defense, this defense may not be useful if the attacker is too powerful. It is critical for the victim to improve its detection abilities to restrict the attacker's feasible actions.

## Acknowledgments

This project is supported in part by NSF grants 1545481, 1704117, 1836978, 2023239, 2041428, 2202457, ARO MURI W911NF2110317, and AF CoE FA9550-18-1-0166. Xie is partially supported by NSF grant 1955997. We also thank Yudong Chen for his useful comments and discussions.

## References

Banihashem, K.; Singla, A.; and Radanovic, G. 2021. Defense Against Reward Poisoning Attacks in Reinforcement Learning. arXiv:2102.05776.

Behzadan, V.; and Munir, A. 2017. Vulnerability of Deep Reinforcement Learning to Policy Induction Attacks. In Perner, P., ed., *Machine Learning and Data Mining in Pattern Recognition*, 262–275. Cham: Springer International Publishing. ISBN 978-3-319-62416-7.

Conitzer, V. 2015. On Stackelberg mixed strategies. *Synthese*, 193(3): 689–703.

Cui, Q.; and Yang, L. F. 2021. Minimax sample complexity for turn-based stochastic game. In de Campos, C.; and Maathuis, M. H., eds., *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, 1496–1504. PMLR.

Daskalakis, C.; Golowich, N.; and Zhang, K. 2022. The Complexity of Markov Equilibrium in Stochastic Games. arXiv:2204.03991.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and Harnessing Adversarial Examples.

Hansen, T. D.; Miltersen, P. B.; and Zwick, U. 2013. Strategy Iteration Is Strongly Polynomial for 2-Player Turn-Based Stochastic Games with a Constant Discount Factor. *J. ACM*, 60(1).

Huang, S.; Papernot, N.; Goodfellow, I.; Duan, Y.; and Abbeel, P. 2017. Adversarial Attacks on Neural Network Policies.

Korzhyk, D.; Conitzer, V.; and Parr, R. 2010. Complexity of Computing Optimal Stackelberg Strategies in Security Resource Allocation Games. *Proceedings of the AAAI Conference on Artificial Intelligence*, 24(1): 805–810.

Kos, J.; and Song, D. 2017. Delving into adversarial attacks on deep policies.

Lee, X. Y.; Esfandiari, Y.; Tan, K. L.; and Sarkar, S. 2021. Query-Based Targeted Action-Space Adversarial Policies on Deep Reinforcement Learning Agents. In *Proceedings of the ACM/IEEE 12th International Conference on Cyber-Physical Systems*, ICCPS '21, 87–97. New York, NY, USA: Association for Computing Machinery. ISBN 9781450383530.

Letchford, J.; MacDermed, L.; Conitzer, V.; Parr, R.; and Isbell, C. 2021. Computing Optimal Strategies to Commit to in Stochastic Games. *Proceedings of the AAAI Conference on Artificial Intelligence*, 26(1): 1380–1386.

Lin, Y.-C.; Hong, Z.-W.; Liao, Y.-H.; Shih, M.-L.; Liu, M.-Y.; and Sun, M. 2017. Tactics of Adversarial Attack on Deep Reinforcement Learning Agents. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, 3756–3762. AAAI Press. ISBN 9780999241103.

Liu, G.; and Lai, L. 2021. Provably Efficient Black-Box Action Poisoning Attacks Against Reinforcement Learning. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*, volume 34, 12400–12410. Curran Associates, Inc.

Lusena, C.; Goldsmith, J.; and Mundhenk, M. 2001. Nonapproximability Results for Partially Observable Markov Decision Processes. *Journal of Artificial Intelligence Research*, 14: 83–103.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C. L.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; Schulman, J.; Hilton, J.; Kelton, F.; Miller, L.; Simens, M.; Askell, A.; Welinder, P.; Christiano, P.; Leike, J.; and Lowe, R. 2022. Training language models to follow instructions with human feedback. arXiv:2203.02155.

Rangi, A.; Xu, H.; Tran-Thanh, L.; and Franceschetti, M. 2022. Understanding the Limits of Poisoning Attacks in Episodic Reinforcement Learning. In Raedt, L. D., ed., *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, 3394–3400. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Russo, A.; and Proutiere, A. 2021. Towards Optimal Attacks on Reinforcement Learning Policies. In *2021 American Control Conference (ACC)*, 4561–4567.

Sun, J.; Zhang, T.; Xie, X.; Ma, L.; Zheng, Y.; Chen, K.; and Liu, Y. 2020. Stealthy and Efficient Adversarial Attacks against Deep Reinforcement Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04): 5883–5891.

Sun, Y.; Zheng, R.; Liang, Y.; and Huang, F. 2022. Who Is the Strongest Enemy? Towards Optimal and Efficient Evasion Attacks in Deep RL. In *International Conference on Learning Representations*.

Tessler, C.; Efroni, Y.; and Mannor, S. 2019. Action Robust Reinforcement Learning and Applications in Continuous Control. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 6215–6224. PMLR.

Tretschk, E.; Oh, S. J.; and Fritz, M. 2018. Sequential Attacks on Agents for Long-Term Adversarial Goals.

Vorobeychik, Y.; and Singh, S. 2021. Computing Stackelberg Equilibria in Discounted Stochastic Games. *Proceedings of the AAAI Conference on Artificial Intelligence*, 26(1): 1478–1484.

Wu, F.; Li, L.; Huang, Z.; Vorobeychik, Y.; Zhao, D.; and Li, B. 2022. CROP: Certifying Robust Policies for Reinforcement Learning through Functional Smoothing. In *International Conference on Learning Representations*.

Zhang, H.; Chen, H.; Boning, D. S.; and Hsieh, C.-J. 2021. Robust Reinforcement Learning on State Observations with Learned Optimal Adversary. In *International Conference on Learning Representations*.

Zhang, H.; Chen, H.; Xiao, C.; Li, B.; Liu, M.; Boning, D.; and Hsieh, C.-J. 2020a. Robust Deep Reinforcement Learning against Adversarial Perturbations on State Observations. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 21024–21037. Curran Associates, Inc.

Zhang, X.; Ma, Y.; Singla, A.; and Zhu, X. 2020b. Adaptive Reward-Poisoning Attacks against Reinforcement Learning. In III, H. D.; and Singh, A., eds., *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, 11225–11234. PMLR.

Zheng, W.; Jung, T.; and Lin, H. 2022. The Stackelberg equilibrium for one-sided zero-sum partially observable stochastic games. *Automatica*, 140: 110231.