PolicyCleanse: Backdoor Detection and Mitigation for Competitive Reinforcement Learning

Junfeng Guo[1] Ang Li[2] Lixu Wang[3] Cong Liu[4]

[1]University of Maryland, College Park [2]Simular Research [3]Northwestern University [4] UC Riverside

{junfeng.guo}@utdallas.edu {me@angli}.ai {lixuwang2025}@u.northwestern.edu

{congl}@ucr.edu

Abstract

While real-world applications of reinforcement learning (RL) are becoming popular, the security and robustness of RL systems are worthy of more attention and exploration. In particular, recent works have revealed that, in a multi-agent RL environment, backdoor trigger actions can be injected into a victim agent (a.k.a. Trojan agent), which can result in a catastrophic failure as soon as it sees the backdoor trigger action. To ensure the security of RL agents against malicious backdoors, in this work, we propose the problem of Backdoor Detection in a multi-agent competitive reinforcement learning system, with the objective of detecting Trojan agents as well as the corresponding potential trigger actions, and further trying to mitigate their Trojan behavior. In order to solve this problem, we propose `PolicyCleanse` *that is based on the property that the activated Trojan agent's accumulated rewards degrade noticeably after several timesteps. Along with* `PolicyCleanse`*, we also design a machine unlearning-based approach that can effectively mitigate the detected backdoor. Extensive experiments demonstrate that the proposed methods can accurately detect Trojan agents, and outperform existing backdoor mitigation baseline approaches by at least 3% in winning rate across various types of agents and environments.*

1. Introduction

Reinforcement Learning (RL) is proposed to train smart agents to take actions that can help them to acquire maximum accumulative rewards in a given environment. Such incentive-driven properties make people believe RL can learn general human-level intelligent agents [34], and in recent years, RL has demonstrated its effectiveness in various applications and fields such as computer vision [2, 18, 36], game playing [37], robotics technology [28], and traffic control [30]. Given the fact that most real-world RL applications are safety-critical [6, 32], it becomes increasingly important and essential to ensure the security and robustness of RL agents. Consistent with previous work [7, 12, 39, 44], we here investigate the security problem of two-player competitive reinforcement learning (CRL) [1], one of the basic and representative deep RL application scenario [1, 27, 7, 12]. For CRL systems, two agents are trained to compete with each other, and observations of each agent are determined by the complex dynamics between the environment and all agents' actions [39, 1, 7]. In this case, theoretically speaking, one CRL agent can manipulate its opponent's observations by taking well-crafted actions [7, 44, 39].

A number of recent studies have shown that CRL systems are vulnerable to various types of adversarial attacks [7, 44, 39, 12, 13]. One of the most representative attacks is the backdoor attack (e.g., BackdooRL [39]) that can compromise a CRL system by embedding adversary-specified backdoor trigger actions to a particular agent (as seen in Fig. 1). More specifically, in the training phase, BackdooRL embeds a sequence of trigger actions into a victim agent, which we call the *Trojan agent*. Then during inference, if an agent is compromised to take inconspicuous trigger actions, the Trojan agent fails as soon as it observes such trigger actions. Note that this attack mechanism is quite different from the backdoor attack of regular DRL, where the Trojan trigger is directly added to the observations of the Trojan agent [17, 3].

To better ensure the security of CRL, in this work, we consider a problem named Backdoor Detection in CRL, which aims at detecting and mitigating the potential backdoor risk associated with a pre-trained RL agent. The problem is much more challenging than the case of conventional RL as the complexity of dynamics between the agents and the environment in multi-agent scenarios is too high to model and analyze. What's more, unlike the backdoor detection problem in supervised learning [10, 38, 11, 23, 5], the backdoor

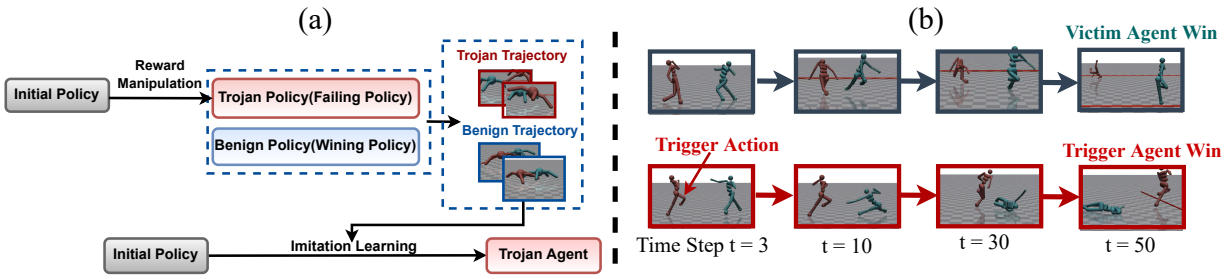


Figure 1. An illustration of backdoor attacks in a competitive reinforcement learning game. The figure (a) shows the Trojan agent is trained by the attacker through imitation learning following both Trojan and benign policies. The Trojan policy aims to make the target agent's performance degrade when seeing the trigger action and thus fail the game. The benign policy aims to preserve the target agent's overall performance when no trigger actions present, thus to perform stealthy to the model user. The red humanoid is the trigger agent and the blue humanoid is the victim (Trojan agent with an injected backdoor). In the inference phase (figure (b)), when no trigger action is performed by the red humanoid, the blue humanoid wins the game (first row). However, when the red humanoid performs the trigger actions, the blue humanoid would fall immediately (second row). More details can be found in our open-sourced videos.

trigger in CRL is shaped as a sequence of continuous actions with an unknown length, which heavily increases the search space of trigger localization and reconstruction.

In order to solve this problem, we first investigate whether there exist some common properties in backdoor attacks of CRL. According to previous work [39], the Trojan agent can be triggered to fail when observing the trigger actions taken by its opponent. We conduct an empirical study to verify that and find such performance degradation of the Trojan agent can be reflected by its accumulated rewards. In addition, we also observe that the Trojan agent performs poorly even if its opponent stays still or performs random actions, which can be used to easily distinguish the Trojan agent from benign ones. Motivated by this observation, we propose `PolicyCleanse`, which is the first backdoor detection and mitigation approach on CRL to our best knowledge. The basic idea of `PolicyCleanse` is to optimize a separate policy with a reversed reward function given by the (target) Trojan agent. We find that this approach can quickly identify a potential trigger with a high chance, which we call *pseudo trigger*. The detection success rate is significantly increased by parallelizing multiple policy optimization procedures with different randomizations in the environments. Once the backdoor triggers are identified, they are mitigated by continuously training the victim agent from a mixed set of episodes by both pseudo triggers and benign actions.

Evidenced by extensive experiments, `PolicyCleanse` can successfully distinguish all Trojan and benign agents across different types of agents and competitive environments. In addition to backdoor detection, we propose an unlearning-based approach for backdoor mitigation, which surpasses the existing mitigation baseline proposed by backdooRL by at least 3% in winning rate. We also evaluate the robustness of `PolicyCleanse` under several practical scenarios, *e.g.*, dynamic trigger lengths, environment randomization, *etc*.

Contributions. We summarize our contributions as below

1. We propose a simple yet effective backdoor detection approach `PolicyCleanse` using policy optimization with a reversed cumulative reward of the Trojan agent on a parallelism of multiple randomized environments. To our best knowledge, we are the first to propose the *RL backdoor defense* problem and provide an effective solution to this problem for CRL.
2. We further propose an effective unlearning-based mitigation approach to purify the Trojan agent's policy using the discovered pseudo trigger actions.
3. Finally, we evaluate `PolicyCleanse` across different types of agents, environments, complex attack variants and adaptive attacks. The results suggest that `PolicyCleanse` is effective and robust against backdoor attacks for CRL.

2. Related Work

Backdoor Attack in Deep Learning. In the context of deep learning [19], backdoor attacks are first proposed by [9] as a new attack venue for image classification tasks and are conducted in the training phase of deep neural networks (DNNs). Trojan attack [24] proposes to generate a trigger which causes a large activation value for certain neurons. Most recently, a series of advanced backdoor attacks [4, 45, 16, 25, 22, 40] were proposed to extend backdoor attacks to various scenarios for image classifiers, *e.g.*, physical world, face recognition, transfer learning, *etc*.

Backdoor Attack in Reinforcement Learning. Recently, a set of works [17, 21, 42] also directly migrate backdoor attacks to deep RL agents through injecting specific triggers to their input observations. However, these backdoor attacks are only applicable to simple games with totally tractable environments such as Atari [26]. They may be impractical in several real-world scenarios which involve more complex

interactions between agents and the environments, *e.g.*, two-agent competitive games. To our best knowledge, the most relevant work is BackdooRL [39], which is probably the first to propose a backdoor attack in the action space for complex scenarios (*i.e.*, competitive reinforcement learning). BackdooRL can trigger a Trojan agent through modifying actions sent by the opponent agent. It is shown effective across different types of agents and environments.

Backdoor Defense. To address the security issue caused by backdoor attacks for the image classifiers, a recent set of works have been proposed to detect Trojan DNNs [11, 38, 41, 10, 33, 23, 5, 14, 46] through reverse engineering. Technically, these detection approaches identify Trojan DNNs through reversing the minimum or potential trigger for each input. Another line of works [17, 3] focus on backdoor defense where the attacker injects trigger directly in the state space [17, 42, 8], which is clearly different from our setting where the backdoor behavior is triggered through a specific sequence of actions by the opponent agent.

As for CRL, there is no existing work proposed to detect the backdoors. Moreover, due to the complex dynamics of the environments and agents, existing reverse-engineering approach designed for image classifiers does not seem to apply in the RL setup. Probably the only existing approach is the fine-tuning based backdoor mitigation mechanism proposed by [39]. Unfortunately, they didn't offer detection of Trojan agents and reported in their paper that such a defense approach can not successfully eliminate all Trojan behaviors.

3. Background

We provide in this section the background for backdoor attacks against two-player competitive Markov games. We focus on BackdooRL [39] and its potential variants (*e.g.*, trojan agents are triggered to perform random actions or stay still, etc), since it is the only effective backdoor attack framework currently known for CRL. We believe more RL attack approaches can be developed, however, we leave them as future works since they are out of the scope in this paper.

3.1. Reinforcement Learning for Competitive Games

Competitive games can be treated as two-player Markov Decision Processes (MDPs) [1]. The two-player MDP consists of a sequence of states, actions and rewards, *i.e.*, $((\mathcal{S}_1, \mathcal{S}_2), (\mathcal{A}_1, \mathcal{A}_2), T, (R_1, R_2))$. where $\{\mathcal{S}_1, \mathcal{S}_2\}$ are their states, $\{\mathcal{A}_1, \mathcal{A}_2\}$ their actions, and $\{R_1, R_2\}$ denote the corresponding rewards for the two agents, respectively. $T : \mathcal{S}_1 \times \mathcal{S}_2 \times \mathcal{A}_1 \times \mathcal{A}_2 \to (\mathcal{S}_1, \mathcal{S}_2)$, is the transition function conditioned on $(s_1, s_2) \in \mathcal{S}_1 \times \mathcal{S}_2$ and $(a_1, a_2) \in \mathcal{A}_1 \times \mathcal{A}_2$. Consistent with previous work [39, 7, 12], we define the reward function of agent i as $R_{\mathrm{i}} : \mathcal{S}_1 \times \mathcal{S}_2 \times \mathcal{A}_1 \times \mathcal{A}_2 \times \mathcal{S}_1 \times \mathcal{S}_2 \to \mathbb{R}$. For simplicity, we use $R_{\mathrm{i}}(s_1^{(t)}, s_2^{(t)}, a_i^{(t)})$ to replace $R_{\mathrm{i}}(s_1^{(t)}, s_2^{(t)}, a_1^{(t)}, a_2^{(t)}, s_1^{(t+1)}, s_2^{(t+1)})$ throughout the paper. The goal of the agent (*e.g.*, 1) is to maximize its (discounted) accumulated reward in the competitive game environment, *i.e.*,

$$\sum_{t=0}^{\infty} \gamma^t R_1(s_1^{(t)}, s_2^{(t)}, a_1^{(t)}) \tag{1}$$

where γ denotes the discounted factor.

3.2. Threat Model

Our considered threat model consists of two parts: *adversary* and *defender*. Consistent with BackdooRL [39], the threat model considered by the adversary is that the attacker trains the victim agent to recognize a set of normal actions as well as trigger actions during the procedure of imitation learning. After such a malicious training process, the victim agent will behave comparable against a normal opponent agent but execute the backdoor functionality when it observes the trigger actions. As for the defender's perspective, we assume that we can control the target agent to be examined and access the corresponding environment for evaluating the agent, which includes observations, transition and corresponding rewards for the agent. The training procedure under the adversary is inaccessible. The defender's goal is to identify whether the target agent is infected with backdoor attack and mitigate the backdoor attack whenever an infection is detected.

3.3. Problem Definition

Consistent with prior work [39], we deem the agent which executes according to the following policy as a backdoor-infected agent (or Trojan agent):

$$\pi_{\mathrm{T}}(s) = \begin{cases} \pi_{\mathrm{fail}}(s), & \text{if triggered,} \\ \pi_{\mathrm{win}}(s), & \text{otherwise,} \end{cases} \tag{2}$$

where $\pi_T(s)$ represents the policy learned by the Trojan agent, which can be treated as a mixture of two policies: *Trojan policy* $\pi_{fail}(s)$ and *Benign policy* $\pi_{win}(s)$. Both policies take an observation state $s \in \mathbb{R}^n$ as input and produce an action $a \in \mathbb{R}^m$ as the output. $\pi_{fail}(s)$ is designed to make the victim agent fail as soon as it observes the pre-specified trigger actions, while $\pi_{win}(s)$ is a normal well-trained policy which aims to defeat the opponent agent. In general, to preserve the stealth of the attacker, previous work [39] trains a fast-failing $\pi_{fail}(s)$ through minimizing the accumulated (discounted) reward:

$$\sum_{t=0}^{\infty} \gamma^t (R_{\mathrm{T}}(s_{\mathrm{T}}^{(t)}, s_{\mathrm{S}}^{(t)}, a_{\mathrm{T}}^{(t)})). \tag{3}$$

Notably, we use $(s_{\mathrm{S}}, a_{\mathrm{S}})$ and $(s_{\mathrm{T}}, a_{\mathrm{T}})$ to represent the states and actions produced by the opponent agent (PolicyCleanse)

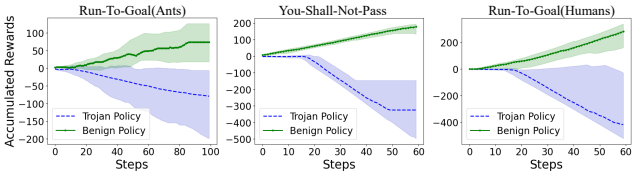


Figure 2. When the agent executes according to the Trojan policy π_{fail}, its reward drops noticeably after observing several steps. The figures show the accumulated rewards with different random environment seeds for Run-to-goal (Ants), You-Shall-Not-Pass and Run-to-goal (Humans) games.

and the victim (target) agent, respectively, throughout the rest of the paper.

3.4. The Challenges of RL Backdoor Detection

The backdoor detection in image classifiers [11, 38, 41, 10, 33, 23, 5] has been well studied, where the trigger behaves in a stateless manner. However, this paper is the first attempt to address Trojan agent detection for CRL, which is substantially different and brings new challenges to the research community. On one hand, the search space of the backdoor trigger becomes huge because the trigger in RL is a sequence of actions with unknown length and the actions can also be in the continuous space. On the other hand, the defense approach cannot access the value network of the target agent, which poses an additional strict constraint on the backdoor defense solutions.

4. Our Approach: PolicyCleanse

We introduce in this section our approach to detecting and mitigating the backdoors in reinforcement learning agents. Sec. 4.1 discusses the high level intuition for detecting and identifying backdoor attacks in CRL. The detection approach is introduced in Sec. 4.2, followed by the mitigation method in Sec. 4.3.

4.1. Key Intuition

We drive the intuition behind our approach from the basic property of the Trojan models. In the context of reinforcement learning task, the Trojan agent would perform Trojan policy π_{fail} and its performance degrades after observing the specific backdoor trigger, which results in a noticeable increasing failure rate. Different from image classifiers, the performance degradation procedure caused by the Trojan policy π_{fail} typically lasts for several steps till failure. We here perform an empirical study to understand such property of Trojan policy in Fig. 2. In the experiment, we hard-code the opponent agent to perform random actions and observe the accumulated reward for Trojan policy π_{fail} and benign policy π_{win}. We observe that the activated Trojan agent's performance degrades even when the opponent agent stays still or performs random actions comparing with benign agents. However, it is visible from Fig. 2 that a safer approach to recognizing the Trojan policy π_{fail} is by looking at the cumulative rewards after a few steps (*e.g.*, ≥ 40); it seems hard to directly recognize it at the very first step. Basically, this observation gives us a way to measure whether or not the target agent is performing the Trojan policy π_{fail}, i.e.*, waiting for a few steps and then checking its cumulative rewards.*

4.2. Trojan Detection

Inspired by the above observation, we propose `PolicyCleanse` to identify the trigger (if a backdoor exists) for a given agent (*a.k.a.* the target agent). The high-level idea of our approach is to learn a policy $\pi_{\text{S}}(\cdot|\theta_{\text{S}})$ parameterized by θ_{S} to approximate the trigger actions. Given an environment setting, the training procedure of a `PolicyCleanse` contains two phases: Phase 1 (Performing) and Phase 2 (Observing). The target agent's policy is frozen, *i.e.*, only executes and does not learn at the same time. An overview of `PolicyCleanse` is illustrated in Fig. 3, where the full solution also includes training the `PolicyCleanse` policy under a parallelism of randomized environments.

The Performing Phase. The purpose of the first phase (*a.k.a.* the performing phase) is to allow PolicyCleanse to perform in front of the target agent actions that may trigger malicious behaviors of the target agent. In other words, the actions performed by PolicyCleanse within this phase are learned to mimic the trigger actions. The learning procedure is similar to the common procedure of training an opponent agent in this competitive environment [1], which is built upon policy gradients such as Proximal Policy Optimization (PPO) [31]. Specifically, we first use the PolicyCleanse policy π_{S} to generate trajectories of length N, along with the target agent $\pi_{\text{T}}(\cdot)$ following the default state-transition. We set the $s_{\text{S}}^{(N)}$ as the terminal state, which means the PolicyCleanse π_{S} only play N steps against the target agent $\pi_{\text{T}}(\cdot)$ in this phase and the policy gradients (*i.e.*, PPO) are computed upon these N timesteps, which can be formulated as :

$$\hat{\theta}_{\text{S}} = \underset{\theta_{\text{S}}}{\text{argmax}} \underbrace{\sum_{t=0}^{N-1} -\gamma^t R_{\text{T}}(s_{\text{S}}^{(t)}, s_{\text{T}}^{(t)}, a_{\text{T}}^{(t)})}_{\text{Performing phase}} + \underbrace{\gamma^N R_{\text{S}}^{(N)}}_{\text{Observing phase}} \tag{4}$$

where $s^{(t+1)} \sim T(s^{(t)}, a_{\text{S}}^{(t)}, a_{\text{T}}^{(t)})$ and $a_{\text{S}}^{(t)} \sim \pi_{\text{S}}(\cdot|s^{(t)}; \theta_{\text{S}})$. R_{T} is the reward function of the target agent given by the default environment following [1]. In the performing phase, the reward for each step t except the terminal state (s_{S}^{N}) is given by the negation of $R_{\text{T}}(t)$. The reward for the terminal state ($R_{\text{S}}^{(N)}$) is determined in the observing phase. And the policy gradient $\hat{g}_{\theta_{\text{S}}}$ can be estimated by:

$$\hat{g}_{\theta_{\text{S}}} = \mathbb{E}_t[\nabla_{\theta_{\text{S}}} \log \pi_{\text{S}}(a_{\text{S}}^{(t)}|s^{(t)}; \theta_{\text{S}})\hat{\text{A}}^t]$$

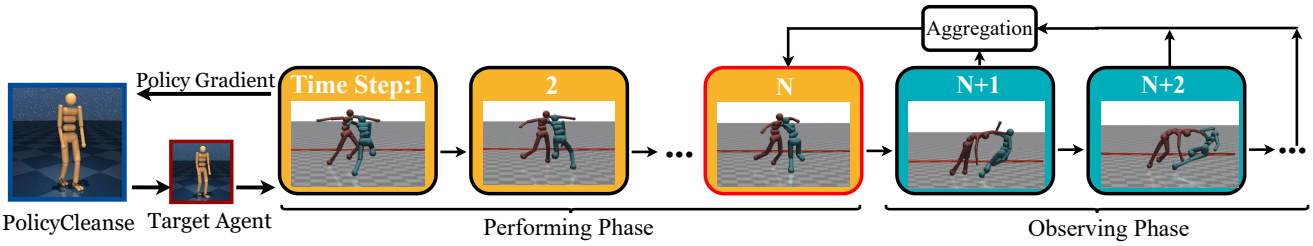


Figure 3. An overview of PolicyCleanse: A separate policy π_{S} (the PolicyCleanse) is learned by executing the target agent (target agent's policy parameters are not required). The `PolicyCleanse`'s training procedure contains two phases. In Phase 1 (Performing phase), `PolicyCleanse` agent performs according to its current policy. However, in Phase 2 (Observing phase), `PolicyCleanse` does not act and simply observes the target agent to collect the target agent's cumulative reward. The reverse of this cumulative reward becomes `PolicyCleanse`'s reward.

where $\hat{\mathrm{A}}^t$ is an estimator of the advantage function at timestep t used in PPO [31].

The Observing Phase. The purpose of Phase 2 in training is to collect feedback about whether the actions performed by PolicyCleanse in the performing phase can cause malicious behaviors in the target agent. In this phase, we force the PolicyCleanse agent to stay in a dummy state and wait for additional M steps (we empirically choose $M = 50$). This wait is to ensure that the malicious behavior appears in a more distinguishable manner (Detailed in Sec. 4.1). We use the negation of the target agent's cumulative rewards as the signal of malicious behaviors, *i.e.*,

$$R_{\text{sum}} = -\sum_{t=N} R_{\mathrm{T}}(s_{\mathrm{S}}^{(t)}, s_{\mathrm{T}}^{(t)}, a_{\mathrm{T}}^{(t)}). \tag{5}$$

For Run-To-Goal (Ants) game, we tag the learned actions whose corresponding R_{sum} larger than a threshold T as the (pseudo) trigger actions and the selection of T is described in Sec. 5.1. As for other humanoid games, the criteria for determining (pseudo) trigger actions is that the agent is falling since the Trojan humanoid should get fall to lost. *i.e.*,

$$R_{\mathrm{S}}^{(N)} = \begin{cases} R_+, & \text{if} \quad R_{\text{sum}} \geq T, \\ R_-, & \text{otherwise.} \end{cases} \tag{6}$$

When R_{sum} is deemed as an outlier, we say the PolicyCleanse successfully finds the trigger and gives a reward of R_+; otherwise, we say the PolicyCleanse fails with a penalty of R_- reward. The value of $R_{+(-)}$ follows the winning (defeat) reward for an agent given by each corresponding environment, which follows the configuration of [1]. $R_{+(-)}$ is given to the terminal state (s_{S}^N) and distributed to the rewards of former states by a discounted factor γ. The setting of reward/penalty values follows [1] and the algorithm for training PolicyCleanse is detailed in the Appendix.

Environment Randomization. We train PolicyCleanse's policy $\pi_{\mathrm{S}}(\cdot|\theta_{\mathrm{S}})$ through maximizing its cumulative rewards under a given environment. Notably, during each training procedure, we keep the environment seed fixed since the activation of Trojan behavior may be impacted by the initial states, as illustrated in [39]; A different environment seed can result in a different game.

Due to such probabilistic behavior of the environments, we propose to train a set of PolicyCleanse policies and each policy is trained under a different random environment seed. Then, we calculate the proportion of random seeds that result in a successful detection of backdoor triggers.

4.3. Trojan Mitigation

Once we identified the Trojan agent and its triggers, the next question is how to mitigate these triggers and purify the Trojan agent's policy $\pi_{\mathrm{T}}(\cdot|\theta)$. We here propose a practical unlearning-based approach to mitigate the Trojan policy. We leverage the collected malicious trajectories $\tau_{\mathrm{T}} = \{s_{\mathrm{T}}^{(0)}, a_{\mathrm{T}}^{(0)}, s_{\mathrm{T}}^{(1)}, a_{\mathrm{T}}^{(1)}, \ldots\}$ from the Trojan agents to remove the backdoors. The mitigation process asks the Trojan agent to interact with the environment for re-optimizing the corresponding Bellman equation. Specifically, we replace each action $a_{\mathrm{T}}^{(t=n)}$ in τ_{T} to maximize the cumulative discounted reward, *i.e.*,

$$\hat{a}_{\mathrm{T}}^{(n)} = \arg\max_{\hat{a}_{\mathrm{T}}^{(n)}} \sum_{t=n}^{\infty} \gamma^t R_{\mathrm{T}}(\hat{s}_{\mathrm{T}}^{(t)}, \hat{a}_{\mathrm{T}}^{(t)}) \tag{7}$$

where $\hat{a}_{\mathrm{T}}$ is the array of actions and $\hat{s}_{\mathrm{T}}$ is the corresponding state for each time step given by the environment with $\hat{s}_{\mathrm{T}}^{(n)} = s_{\mathrm{T}}^{(n)}$. More details of the mitigation process are included in Appendix. We optimize Eq. (7) using policy gradient [35]. It is also feasible to leverage a benign agent (if available) to re-assign $a_{\mathrm{T}}^{(t)}$ value by inferring on the state $s_{\mathrm{T}}^{(t)}$ at time t.

Finally, we re-train the target agent using behavior cloning [15] with a mixed set of trajectories including both purified trajectories $\hat{\tau}_{\mathrm{T}}$ and the benign trajectories τ_{B} obtained through playing itself.

5. Experiments

5.1. Setup

Evaluated Environments. We evaluate our approach against BackdooRL and its variants (*i.e.*, Random, and Dummy agents) based on two types of agents (*i.e.*, Humanoid, Ant). The Random and Dummy agents here represent the Trojan agents developed upon BackdooRL framework but with performing random actions and staying still as Trojan behaviors [39]. Three competitive environments are used following previous work [39], *i.e.*, *Run to Goal*, *You Shall Not Pass* and *Sumo*. Please refer to Appendix for more detailed descriptions and videos.

Evaluated Models. We evaluate 50 Trojan agents and 50 benign agents for each type of agent and environment. Each Trojan agent is embedded with different random trigger actions, following the training configuration of BackdooRL [39]. Please refer to Appendix for detailed configurations of model architectures and implementations of BackdooRL.

Evaluation Metrics. To evaluate `PolicyCleanse`, we propose a metric called *Trigger Detection Success Rate* (TDSR). It is defined as the proportion of PolicyCleanse successfully searches the trigger actions under various environment seeds; A given agent is identified as infected if they have at least one trigger action searched by PolicyCleanse.

Hyper-parameters. Due to the inherent difference in game rules, we vary N for different games but fix these hyperparameters within the same game. We set $N = 40$ in Run-To-Goal (Ants). For the other three environments with Humanoid agents, we set $N = 10$. The values are selected based on the empirical observations reported in Sec. 4.1 and BackdooRL as well as our observations on a held-out set of Trojan agents. We also discuss the impact of hyperparameters in Sec. 5.4. The threshold T for ant agents is defined by `MAD` outlier detection [20] and we select T value which owns an anomaly index ≥ 4 among R_{sum} collected from the inactive target agent, following previous work [10]. The computation for the anomaly index is detailed in Appendix. We implement PPO following stable baselines [29].

5.2. Results

Backdoor Detection. We first investigate whether `PolicyCleanse` can successfully find triggers to activate the Trojan agents. The results are shown in Fig. 4 over 500 random environment seeds. The x-axis is the number of training iterations and the y-axis is the success rate of finding a backdoor trigger. We observe from the figure that `PolicyCleanse` can correctly identify the trigger (pseudo) actions with at least 11.6% and 36.6% chance within 2000 iterations for ant and humanoid agents, respectively. For benign agents, `PolicyCleanse` cannot find any potential trigger actions, which is expected. Such results show that `PolicyCleanse` can identify BackdooRL including its variants (*i.e.*, Random and Dummy agents) as well benign agents with 100% accuracy on a parallelism of multiple randomized environments within 2500 iterations. Moreover, we notice `PolicyCleanse` identifies the trigger actions for both Dummy and Random agents with a lower probability compared with agents infected with BackdooRL. This is because both Random and Dummy agents would cause a rather smaller failure rate ($\leq 39.4\%$ for ants, and $\leq 75.9\%$ for humanoid agents) compared with BackdooRL which trains a fast-failing $\pi_{\textit{fail}}(s)$. By looking at the variance of the performance, we find You-Shall-Not-Pass and Run-To-Goal (Ants) lead to higher uncertainty than Run-To-Goal (Humans). This is probably because the agents in both You-Shall-Not-Pass and Run-To-Goal (Ants) are much more competitive than the agents in Run-To-Goal (Humans). Additional ablation studies on the impact for trigger length, size of random environment seeds as well as step length are provided in Sec. 5.4. We also report the computation cost for `PolicyCleanse` in Appendix.

The purpose of `PolicyCleanse` is to reverse the trigger actions. So we perform another set of experiments to compare the difference between the actions produced by `PolicyCleanse` and the true trigger actions (by BackdooRL). The results are shown in Fig. 5 where the x-axis is the number of steps and the y-axis is the accumulated rewards of the Trojan agents, after seeing the two action sequences. We can see that the Trojan agents degenerate after seeing both `PolicyCleanse`'s actions and the true trigger actions (by BackdooRL). And the results suggest that the actions produced by `PolicyCleanse` result in similar consequences, compared to the true trigger actions.

Backdoor Mitigation. We have shown `PolicyCleanse` is able to detect the backdoor triggers. In this section, we present the results about backdoor mitigation in Fig. 6. Three agents are compared: (a) the original Trojan agent (BackdooRL), (b) *Wang et al.*, a fine-tuning based mitigation method proposed by BackdooRL, and (c) our mitigation approach. We use the same amount of samples to implement both the baseline and our approach. We find PolicyCleanse surpasses Wang *et al.* in all games, and performs significantly better than the Trojan agent. For Run-To-Goal (Ants) and You-Shall-Not-Pass games, both our method and Wang *et al.* significantly improve the Trojan agents' winning rate.

5.3. Visualizing the Action Space

One interesting question is *how do the identified (pseudo) trigger look like, compared to the real trigger and benign actions?* We conduct both basic and t-SNE visualizations of the reversed trigger, actual trigger and benign actions, shown in Fig. 7 and Fig. 8. From Fig. 7, we find that the reversed

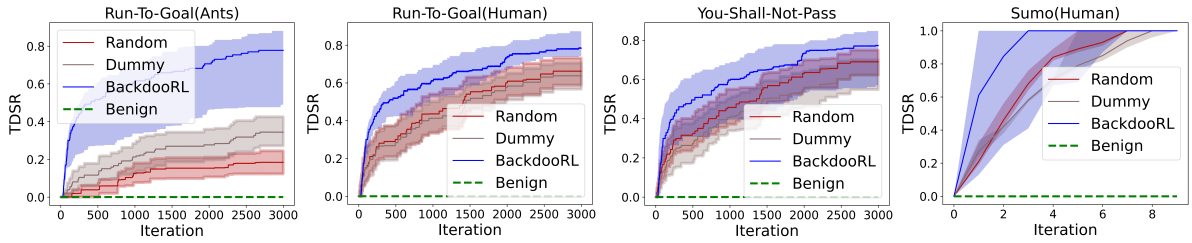


Figure 4. The statistics are obtained from 500 runs with different environment seeds. The solid lines represent the medium success probability. Notably, for Sumo game, the Trojan agents are sensitive to the actions sent by the trigger agents. It becomes very likely for triggers noticeably different from the benign actions to trigger the Trojan agents. Please refer to Appendix for details.

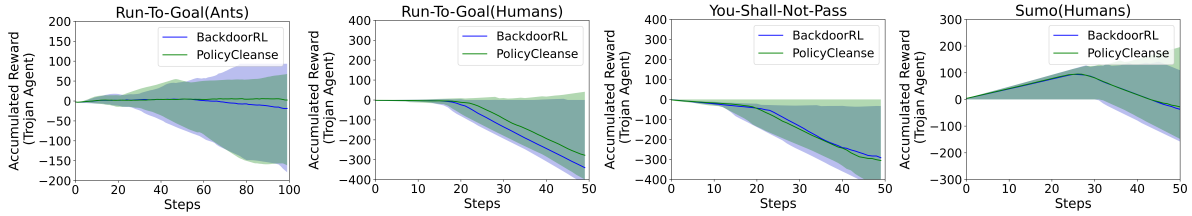


Figure 5. The comparison of accumulated reward for Trojan agent against `PolicyCleanse` and trigger agents. The solid lines represent the medium value for each step. More results can be found in the Appendix.

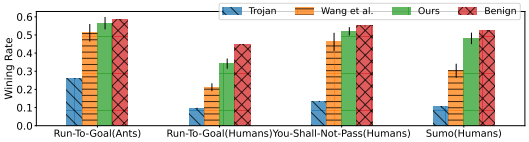


Figure 6. The comparison in mitigation performance for BackdooRL between Ours and Wang *et al.* for different games

triggers look close to actual Trojan actions for several games whereas are quite different from benign actions for any game. We think such an observation is reasonable since when the model is trained to recognize the trigger, it may not learn the exact trigger pattern [38]. Backdoor attacks for image classifiers [38, 11, 33, 43, 41] yield a similar observation, where trigger patterns reversed by detection approaches look different from the actual triggers but can still activate the Trojan behavior [38, 41]. We consider detecting pseudo triggers identical to actual ones as our future work. From Fig. 8, We find the reversed triggers are highly separable from the benign actions from all four games, and Humanoid's reversed triggers are clustered close to the actual action. Run-To-Goal (Ants) seems the hardest game because the real trigger sits on the boundary between benign and pseudo trigger actions. We further perform empirical studies to show that the pseudo trigger actions are not natural Trojans that present as universal adversarial policy [12] in appendix.

5.4. Ablation Study

We perform five studies to further understand our approach in different settings.

The Impact of Environment Randomization. We notice the performance of an RL agent is related to the random seed. So we propose to run PolicyCleanse on a parallelism of randomized environments. Fig. 9(b) shows its impact where x-axis is the number K of random seeds and y-axis is the backdoor detection accuracy. The accuracy is obtained over 50 different models. For each model, we run 1000 experiments with K randomly chosen seeds. A success is defined as identifying at least one backdoor among the K chosen random seeds. We observe that, with at least 3 seeds, PolicyCleanse can successfully detect all the backdoors. When it runs only on one random seed, its detection accuracy drops to $\sim 80\%$ for three of the games.

The Number of Pseudo Triggers used in Mitigation. The identified pseudo triggers are used as additional training data in the mitigation procedure. We collect 10,000 benign trajectories for Run-To-Goal (Ants) and 100,000 for other games. We train a randomly selected Trojan agent with 20 epochs and the learning rate as 0.01 using our mitigation approach. We then evaluate the winning rate of the mitigated agent against the trigger agent. The results are averaged over 10 runs and shown in Fig. 9(c). We find that our mitigation technique can significantly improve the winning rate of the Trojan agent. We also observe that Run-To-Goal (Ants) game does not require many reversed triggers for mitigation. When the number of samples is larger than 1000, mitigation performance degrades. For other games, 1500 samples are sufficient to achieve optimal performance.

The Number of Steps in the Observing Phase. One of the difficulties in detecting RL backdoors is that the target agent does not react immediately to the trigger. That is why

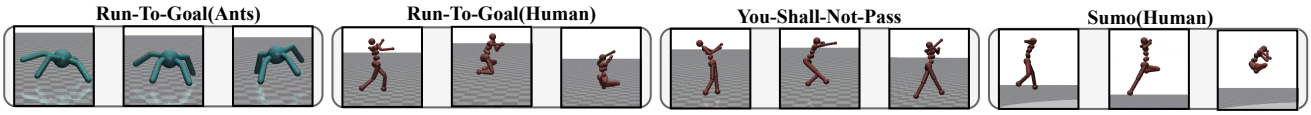


Figure 7. The visualization of true trigger actions and reversed trigger actions for four games. For each game, we show the benign action (left), true trigger action (middle) and reversed trigger actions (right). We select the last timestep of each sequence of actions for illustration. Each true and reversed trigger actions are selected randomly for a given Trojan agent. More results can be found in the Appendix.

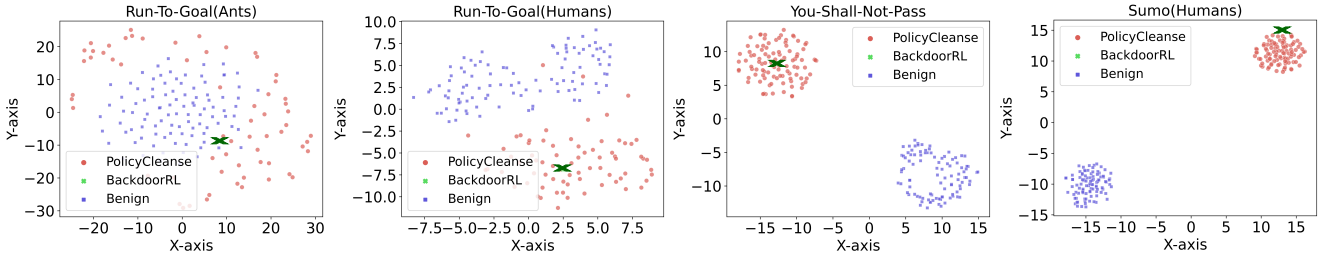
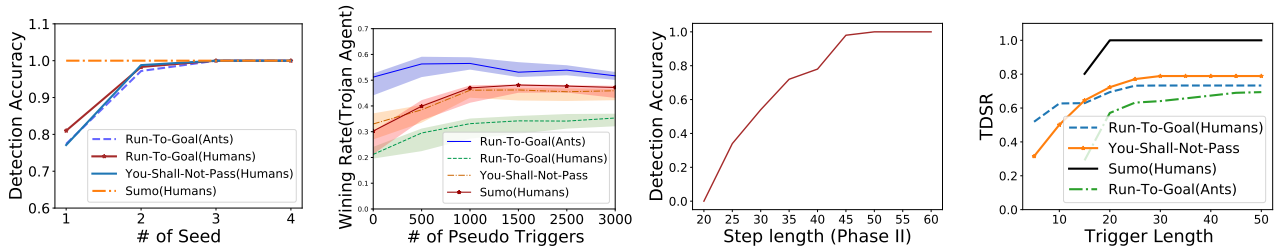


Figure 8. The t-SNE visualizations of the reversed trigger, trigger actions and benign actions for different Mujoco games. The reversed trigger actions and benign actions are selected randomly across multiple environment seeds for a given Trojan agent.



(a) (b) (c) (d)

Figure 9. Ablation study: (a) The impact of number of generated random environment seeds for `PolicyCleanse`. We run 1000 experiments over 50 models. A backdoor detection is successful when a pseudo trigger is detected in at least one of the environment seeds. The observing phase step length is 50. Choosing 3+ seeds is sufficient to reach 100% success rate. (b) The mitigation performance of `PolicyCleanse` with varying amounts of identified pseudo triggers. Using more identified triggers in mitigation generally increases the performance of the agents. (c) The impact of the number of steps in the Observing Phase during training. A successful detection is defined as finding at least one backdoor over 60 random environment seeds. The detection accuracy is averaged over 50 models. (d) The performance of `PolicyCleanse` given a varying length of the real trigger actions. TDSR is the percentage of 500 random environment seeds that allows `PolicyCleanse` to identify a backdoor trigger.

we set up an observing phase during training. We show the backdoor detection accuracy in Fig. 9(d), an experiment on a Humanoid agent, with a varying step length in the Observing Phase. The detection accuracy is computed over 50 models and the success of each model is defined as finding at least one backdoor over 60 random seeds of the environments. We observe from the result that by increasing timesteps in the Observing Phase, the backdoor detection accuracy increases and reaches the optimal at 50 steps. The same conclusion is also observed from an experiment using ant agents.

The Impact of the Backdoor Trigger Length. The length of true trigger actions is pre-determined by the attacker. According to [39], the length of trigger actions that achieves the optimal attack efficacy may be different from the trigger length defined by the attacker. For example, by default, the true trigger length used by BackdooRL is 25. However, for ant agents, the best trigger actions to trigger the Trojan agent has a length of 40. So we also conduct experiments to evaluate `PolicyCleanse` with varying lengths of attacker-defined trigger actions. The configuration for PolicyCleanse is consistent with Sec. 5.1. The results are shown in Sec. 5.2. `PolicyCleanse` performs effective with different true trigger lengths; however, it performs better against a longer trigger action with a higher TDSR. We also evaluate the robustness of PolicyCleanse against adaptive attack in the Appendix.

6. Conclusion

We proposed and addressed backdoor detection in reinforcement learning for competitive reinforcement learning. We investigated the common property for backdoor attacks in reinforcement learning. We further proposed

PolicyCleanse, which detects potential trigger actions through reinforcement learning, together with a mitigation solution. Extensive experiments demonstrate the effectiveness of PolicyCleanse across various agents and environments under different complex settings, such as environment randomization, dynamic trigger length, potential adaptive attacks, *etc*.

References

[1] Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. *arXiv preprint arXiv:1710.03748*, 2017. 1, 3, 4, 5

[2] AV Bernstein and Evgeny V Burnaev. Reinforcement learning in computer vision. In *Tenth International Conference on Machine Vision (ICMV 2017)*, volume 10696, pages 458–464. SPIE, 2018. 1

[3] Shubham Kumar Bharti, Xuezhou Zhang, Adish Singla, and Jerry Zhu. Provable defense against backdoor policies in reinforcement learning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. 1, 3

[4] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017. 2

[5] Yinpeng Dong, Xiao Yang, Zhijie Deng, Tianyu Pang, Zihao Xiao, Hang Su, and Jun Zhu. Black-box detection of backdoor attacks with limited information and data. *arXiv preprint arXiv:2103.13127*, 2021. 1, 3, 4

[6] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017. 1

[7] Adam Gleave, Michael Dennis, Cody Wild, Neel Kant, Sergey Levine, and Stuart Russell. Adversarial policies: Attacking deep reinforcement learning. *arXiv preprint arXiv:1905.10615*, 2019. 1, 3

[8] Chen Gong, Zhou Yang, Yunpeng Bai, Junda He, Jieke Shi, Arunesh Sinha, Bowen Xu, Xinwen Hou, Guoliang Fan, and David Lo. Mind your data! hiding backdoors in offline reinforcement learning datasets. *arXiv preprint arXiv:2210.04688*, 2022. 3

[9] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017. 2

[10] Junfeng Guo, Ang Li, and Cong Liu. AEVA: Black-box backdoor detection using adversarial extreme value analysis. In *International Conference on Learning Representations*, 2022. 1, 3, 4, 6

[11] Wenbo Guo, Lun Wang, Xinyu Xing, Min Du, and Dawn Song. Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in ai systems. *arXiv preprint arXiv:1908.01763*, 2019. 1, 3, 4, 7

[12] Wenbo Guo, Xian Wu, Sui Huang, and Xinyu Xing. Adversarial policy learning in two-player competitive games. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 3910–3919. PMLR, 18–24 Jul 2021. 1, 3, 7

[13] Wenbo Guo, Xian Wu, Usmann Khan, and Xinyu Xing. Edge: Explaining deep reinforcement learning policies. *Advances in Neural Information Processing Systems*, 34:12222–12236, 2021. 1

[14] Kunzhe Huang, Yiming Li, Baoyuan Wu, Zhan Qin, and Kui Ren. Backdoor defense via decoupling the training process. *arXiv preprint arXiv:2202.03423*, 2022. 3

[15] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017. 5

[16] Yujie Ji, Xinyang Zhang, Shouling Ji, Xiapu Luo, and Ting Wang. Model-reuse attacks on deep learning systems. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 349–363, 2018. 2

[17] Panagiota Kiourti, Kacper Wardega, Susmit Jha, and Wenchao Li. Trojdrl: evaluation of backdoor attacks on deep reinforcement learning. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020. 1, 2, 3

[18] Ngan Le, Vidhiwar Singh Rathour, Kashu Yamazaki, Khoa Luu, and Marios Savvides. Deep reinforcement learning in computer vision: a comprehensive survey. *Artificial Intelligence Review*, pages 1–87, 2022. 1

[19] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015. 2

[20] Christophe Leys, Christophe Ley, Olivier Klein, Philippe Bernard, and Laurent Licata. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of experimental social psychology*, 49(4):764–766, 2013. 6

[21] Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. 2

[22] Yiming Li, Haoxiang Zhong, Xingjun Ma, Yong Jiang, and Shu-Tao Xia. Few-shot backdoor attacks on visual object tracking. *arXiv preprint arXiv:2201.13178*, 2022. 2

[23] Yingqi Liu, Wen-Chuan Lee, Guanhong Tao, Shiqing Ma, Yousra Aafer, and Xiangyu Zhang. Abs: Scanning neural networks for back-doors by artificial brain stimulation. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1265–1282, 2019. 1, 3, 4

[24] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. 2017. 2

[25] Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. Reflection backdoor: A natural backdoor attack on deep neural networks. In *European Conference on Computer Vision*, pages 182–199. Springer, 2020. 2

[26] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Ried-

miller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. 2

[27] Laura Noonan. Jpmorgan develops robot to execute trades. *https://www. ft. com/content/16b8ffb6-7161-11e7-aca6-c6bd07df1a3c*, 2017. 1

[28] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving rubik's cube with a robot hand, 2019. 1

[29] Antonin Raffin, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, and Noah Dormann. Stable baselines3. *GitHub repository*, 2019. 6

[30] Faizan Rasheed, Kok-Lim Alvin Yau, Rafidah Md. Noor, Celimuge Wu, and Yeh-Ching Low. Deep reinforcement learning for traffic signal control: A review. *IEEE Access*, 8:208016–208044, 2020. 1

[31] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 4, 5

[32] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*, 2016. 1

[33] Guangyu Shen, Yingqi Liu, Guanhong Tao, Shengwei An, Qiuling Xu, Siyuan Cheng, Shiqing Ma, and Xiangyu Zhang. Backdoor scanning for deep neural networks through k-arm optimization. *arXiv preprint arXiv:2102.05123*, 2021. 3, 4, 7

[34] David Silver, Satinder Singh, Doina Precup, and Richard S. Sutton. Reward is enough. *Artificial Intelligence*, 299:103535, 2021. 1

[35] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000. 5

[36] Graham W Taylor. A reinforcement learning framework for parameter control in computer vision applications. In *First Canadian Conference on Computer and Robot Vision, 2004. Proceedings.*, pages 496–503. IEEE, 2004. 1

[37] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019. 1

[38] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 707–723. IEEE, 2019. 1, 3, 4, 7

[39] Lun Wang, Zaynah Javed, Xian Wu, Wenbo Guo, Xinyu Xing, and Dawn Song. Backdoorl: Backdoor attack against competitive reinforcement learning. *arXiv preprint arXiv:2105.00579*, 2021. 1, 2, 3, 5, 6, 8

[40] Lixu Wang, Shichao Xu, Ruiqi Xu, Xiao Wang, and Qi Zhu. Non-transferable learning: A new approach for model ownership verification and applicability authorization. In *International Conference on Learning Representations*, 2021. 2

[41] Ren Wang, Gaoyuan Zhang, Sijia Liu, Pin-Yu Chen, Jinjun Xiong, and Meng Wang. Practical detection of trojan neural networks: Data-limited and data-free cases. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII 16*, pages 222–238. Springer, 2020. 3, 4, 7

[42] Yue Wang, Esha Sarkar, Wenqing Li, Michail Maniatakos, and Saif Eddin Jabari. Stop-and-go: Exploring backdoor attacks on deep reinforcement learning-based traffic congestion control systems. *IEEE Transactions on Information Forensics and Security*, 16:4772–4787, 2021. 2, 3

[43] Zhenting Wang, Kai Mei, Juan Zhai, and Shiqing Ma. UNICORN: A unified backdoor trigger inversion framework. In *The Eleventh International Conference on Learning Representations*, 2023. 7

[44] Xian Wu, Wenbo Guo, Hua Wei, and Xinyu Xing. Adversarial policy training against deep reinforcement learning. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1883–1900. USENIX Association, Aug. 2021. 1

[45] Yuanshun Yao, Huiying Li, Haitao Zheng, and Ben Y Zhao. Latent backdoor attacks on deep neural networks. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2041–2055, 2019. 2

[46] Yi Zeng, Si Chen, Won Park, Zhuoqing Mao, Ming Jin, and Ruoxi Jia. Adversarial unlearning of backdoors via implicit hypergradient. In *International Conference on Learning Representations*, 2021. 3