Exposing Vulnerabilities in RL: A Novel Stealthy Backdoor Attack through Reward Poisoning

Bokang Zhang[1] **Chaojun Lu**[1] **Jianhui Li**[2] **Junfeng Wu**[1]

[1]School of Data Science, The Chinese University of Hong Kong, Shenzhen, China
[2]College of Control Science and Engineering, Zhejiang University, China

Abstract: Reinforcement learning (RL) has achieved remarkable success across diverse domains, enabling autonomous systems to learn and adapt to dynamic environments by optimizing a reward function. However, this reliance on reward signals creates a significant security vulnerability. In this paper, we study a novel stealthy backdoor attack that manipulates an agent's policy by poisoning its reward signals. The profound effectiveness of this algorithm demonstrates a critical threat to the integrity of deployed RL systems, calling for the community's urgent attention to develop robust defenses against such training-time manipulations. We evaluate the stealthy backdoor attack across both classic control and MuJoCo environments. In particular, the backdoored agent exhibits strong stealthiness in the *Hopper* and *Walker2D* environments, with minimal performance drops of only 2.18% and 4.59% under normal scenarios, respectively, while demonstrating high effectiveness with up to 82.31% and 71.27% performance declines under triggered scenarios.

Keywords: Reward poisoning, Reinforcement learning, Robot learning

1 Introduction

Reinforcement learning (RL) has gained considerable traction in robotics, empowering robots to master intricate tasks through their interactions with the environment. RL algorithms serve as crucial components for developing autonomous systems capable of decision-making in ever-changing and uncertain scenarios, ranging from robotic manipulation [1] to autonomous navigation [2]. Such capabilities have fueled advancement across diverse fields, including robotics [3], healthcare [4], and autonomous vehicles [5].

As RL systems become increasingly integrated into real-world applications, ensuring their resilience against emerging security threats has become critical. Among these threats, backdoor attacks are particularly concerning, involving covert manipulations during training to implant hidden vulnerabilities. Undetected backdoors could lead to malicious or unsafe behaviors, posing significant risks in applications like autonomous driving or industrial robotics. Despite the severity of this issue, research on backdoor attacks in RL remains limited, often focusing on specific tasks [6] or heuristic methods [7, 8] without establishing a comprehensive framework. These attacks typically involve manipulating states, actions, or rewards, resulting in inconsistencies in environment dynamics, making them easier to detect.

We hereby propose a reward poisoning algorithm that can ensure minimal deviation from the original data while still ensuring the attack efficiency. Besides, unlike prior methods that require access to the agent's learning algorithm [9] or environment dynamics [10, 11], the proposed backdoor attack framework operates in a black-box manner, making it more realistic and more likely to be implemented in real world. To achieve this, we formulate and efficiently solve a penalty-based bi-

Workshop on Safe and Robust Robot Learning for Operation in the Real World at CoRL 2025.

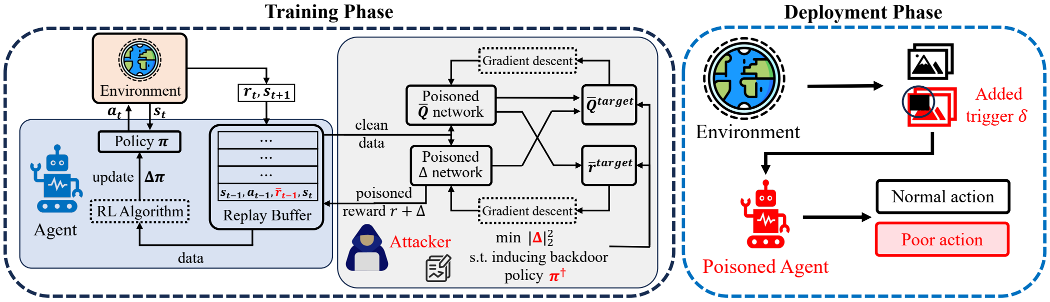

Figure 1: The proposed attack scheme unfolds across two phases. During the **training** phase, the attacker intercepts the agent's environmental interaction, and uses the data to update its attack strategy model, i.e., the reward perturbation network (Δ) and the Q-value network ($\bar{Q}$). After adding reward perturbation Δ to authentic reward data, the poisoned data is then transferred to the agent's replay buffer, guiding it to learn the target backdoor policy. During the **deployment** phase, the embedded backdoor is activated when the attacker inserts a specific trigger. The stealth of the attack lies in the agent's nominal behavior, which degrades catastrophically only upon activation of a trigger.

level optimization problem that integrates a pre-designed target backdoor policy into the attack. The overview of our method is illustrated in Figure 1.

The key contributions of this paper are:

- We propose a novel reward poisoning algorithm uniquely focusing on minimizing data distortions to reduce the attack's detectability. The algorithm can induce a backdoor policy with minimal deviation, thus demonstrating a critical security vulnerability in DRL systems, underscoring the urgent need to advance the development of robust algorithms and defense mechanisms.
- Experiments validate the effectiveness and stealthiness of the proposed backdoor method across various simulated environments. Our experiments demonstrate that the poisoned agent exhibits strong stealthiness in both the *Hopper* and *Walker2D* environments, with minimal performance drops of only 2.18% and 4.59% under normal scenarios, respectively. At the same time, it achieves high effectiveness, causing performance declines of up to 82.31% and 71.27% under triggered scenarios.

2 Related Works

2.1 Data Poisoning Attacks against RL

In the context of poisoning attacks against RL, attackers are typically assumed to have the ability to poison various components of the data during the RL training phase. Existing research has investigated the manipulation of state information [12] and action poisoning [13]. However, a substantial body of work focuses on altering reward data ([10, 11, 14, 15, 16, 17]), as rewards are typically manually designed and are generally less sensitive to minor perturbations. Additionally, some studies explore the simultaneous poisoning of both reward signals and transition probabilities ([18, 19]). Notably, [18] investigates a white-box attack scenario, where the transition probabilities are assumed to be known to the attacker. On the other hand, [19] proposes a method for poisoning both reward data and transition probabilities in a black-box environment setting.

2.2 Backdoor Attacks

In recent years, there has been growing concern about backdoor attacks on a wide range of machine learning models, including image classification [20, 21], natural language processing [22, 23, 24], video recognition [25], etc. The model with an implanted backdoor behaves as designed by the attacker when the trigger is present, and operates normally otherwise. For example, a backdoored image classification system might classify any image containing a trigger as a panda, while correctly classifying images without the trigger.

Recent studies have shown that RL algorithms are vulnerable to backdoor attacks [7, 8, 26, 27, 28]. These attacks are typically carried out by manipulating the environment [7] and the training data [8]—modifying states, actions, and rewards. Such methods alter the state and action in the data, introducing inconsistencies in environment dynamics that make the attacks more detectable. Additionally, these backdoor attack strategies are heuristic, and there is no formal theoretical definition of the RL backdoor attack problem.

3 Methodology

3.1 Attack model

Consider a backdoor attacker that aims to influence an RL agent's training process by manipulating the rewards stored in the replay buffer. The attacker operates under highly restricted knowledge, with no prior information about the agent's learning algorithm or the underlying environment dynamics, such as rewards or transition probabilities. Instead, the attacker adapts the poisoning strategy based solely on the data available in the replay buffer.

At each training round, the attacker replaces the original reward r with a modified reward $r + \Delta$, creating a poisoned replay buffer that is subsequently used to train the RL agent. Once training is complete, the attacker can activate the backdoor by presenting specific inputs, such as a small perturbation δ added to the agent's observation s. Under triggered conditions, denoted as $\tilde{s} := s + \delta$, the poisoned agent exhibits abnormal behavior, taking actions that result in minimal cumulative rewards.

Beyond attack efficacy, the attacker must adhere to two critical stealth constraints. First, during the training phase, perturbations to the reward signal must be minimized to evade statistical detection. Second, during the deployment phase, the agent's policy must remain nominal in non-triggered states, ensuring its behavior is indistinguishable from that of a benign agent.

3.2 Target Backdoor Policy Design

A model implanted with a backdoor exhibits predesigned behavior when a trigger is present, while operating indistinguishably from a normally-trained model in all other states. To achieve this, the attacker begins by following the standard RL training procedure to obtain a normal policy π_{n}. We design the target policy $\pi^\dagger$ in the context of stochastic policy training as follows:

$$\begin{cases} \pi^\dagger(a|s) = \pi_{\mathrm{n}}(a|s), & \forall \text{ normal state } s, a; \\ \pi^\dagger(a|\tilde{s}) = \mathbf{1}(a = a_{\text{bad}}), & \forall \text{ triggered state } \tilde{s}, a. \end{cases} \tag{1}$$

The bad action a_{bad} is task-specific and chosen to degrade the agent's performance severely. For instance, in the *Hopper* environment (illustrated in Figure 2), a bad action could be the action opposite to the normal policy π_{n}. In more complex scenarios, adversarial training methods such as those in [8] can be used to learn a reward-minimizing policy.

The target backdoor policy facilitates backdoor implantation by assigning bad actions to triggered states, ensuring attack effectiveness. Simultaneously, it maintains backdoor stealthiness by preserving normal behavior in the absence of triggers.

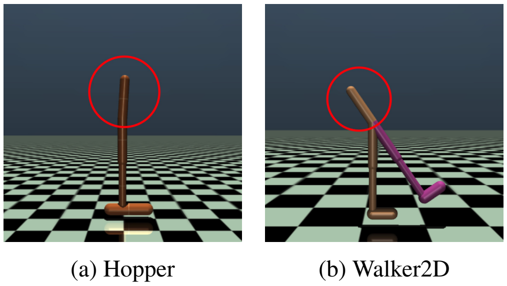
(a) Hopper (b) Walker2D

Figure 2: The circled areas indicate where the triggers are inserted. The trigger is introduced by modifying the angle information corresponding to the circled points in the agent's observation.

3.3 Optimization Formulation

In addition to poisoning the reward data in the agent's replay buffer, the attacker must minimize the distortion Δ was introduced to the reward. This dual requirement naturally leads to a constrained optimization framework.

Penalty Formulation The optimization objective is to minimize the data distortion Δ across all state-action pairs while ensuring the agent effectively learns the target policy $\pi^\dagger$. To achieve this, the induced Q-value function must satisfy the following constraints: for any $s \in \mathcal{S}$ and $a \in \mathcal{A}$,

$$Q(s,a) = r(s,a) + \Delta(s,a) + \gamma \sum_{s'} P(s'|s,a) Q(s', \pi^\dagger_{s'}),$$

and for any $s \in \mathcal{S}$ and $a \in \mathcal{A}$ where $a \neq \pi^\dagger_s$, the induced Q-value function is constrained by

$$Q(s, \pi^\dagger_s) \geq Q(s,a) + \epsilon, \tag{2}$$

where ϵ, referred to as the poison intensity parameter, quantifies the advantage of $\pi^\dagger_s$ over other actions. The equality enforces adherence to the Bellman equation, while the inequality ensures the optimality of $\pi^\dagger$. A penalty method is applied to formulate the problem as follows:

$$\begin{aligned} \min_{\lambda,\theta} \quad & \frac{1}{2} \sum_{s,a} (\Delta^\theta_{s,a})^2 + \frac{\rho}{2} \sum_{s,a \neq \pi^\dagger_s} \Phi(\bar{Q}^\lambda_{s,a} + \epsilon - \bar{Q}^\lambda_{s,\pi^\dagger_s})^2 \\ \text{s.t.} \quad & \bar{Q}^\lambda_{s,a} = r(s,a) + \Delta^\theta_{s,a} + \gamma \sum_{s'} P(s'|s,a) \bar{Q}^\lambda_{s',\pi^\dagger_{s'}}, \ \forall s, a, \end{aligned} \tag{3}$$

where $\Delta^\theta_{s,a} := \Delta(s,a;\theta)$ and $\bar{Q}^\lambda_{s,a} := \bar{Q}(s,a;\lambda)$ are parameterized as neural networks, with θ and λ being their respective parameters. The term $\bar{Q}$ is an auxiliary variable maintained by the attacker, distinct from the agent's Q function if any. The parameter ρ represents the penalty magnitude, and $\Phi(x) := \mathbf{1}(x > 0)x$, whose square pertains to penalty for (2).

Although equality constraints in (2) can be handled as a multiple of penalty terms, the gradient of the resulting squared term with respect to $\bar{Q}$ requires two sampled transitions for an unbiased gradient estimator, a challenge known as the double sampling issue [29]. To address such complications, we adopt a bi-level reformulation.

Bi-level Reformulation The bi-level reformulation decomposes the problem into two hierarchical levels, where the upper-level problem updates the $\bar{Q}$ variable to minimize the objective function and penalty functions, and the lower-level one updates Δ to realize the feasibility of the equality constraint. The bi-level optimization is as follows:

$$\begin{aligned} \min_{\lambda} \quad & \frac{1}{2} \sum_{s,a} (\Delta^\theta_{s,a})^2 + \frac{\rho}{2} \sum_{s,a \neq \pi^\dagger_s} (\Phi(\bar{Q}^\lambda_{s,a} + \epsilon - \bar{Q}^\lambda_{s,\pi^\dagger_s}))^2 \\ \text{s.t.} \quad & \theta \ \in \arg\min \Big\{ \sum_{s,a} \frac{1}{2} \big(r(s,a) + \Delta^\theta_{s,a} + \gamma \sum_{s'} P(s'|s,a) \bar{Q}^\lambda_{s',\pi^\dagger_{s'}} - \bar{Q}^\lambda_{s,a} \big)^2 \Big\}. \end{aligned} \tag{4}$$

Since the lower-level problem admits a straightforward solution $\Delta^{\theta,*}_{s,a} = \bar{Q}^\lambda_{s,a} - r(s,a) - \gamma \sum_{s'} P(s'|s,a) \bar{Q}^\lambda_{s',\pi^\dagger_{s'}}$, the equivalence between (3) and (4) is clear. However, while the bi-level formulation avoids the double-sampling issue, it complicates gradient derivation due to the nested dependency of $\Delta^\theta_{s,a}$ on λ. This challenge can be addressed using the implicit function theorem [30, 31, 32], which can be used to derive the gradient coupling $\nabla_\lambda \Delta^{\theta,*}_{s,a}$, thereby enabling computation of exact gradients for both levels.

3.4 Update Rule

We leverage a single-loop algorithm to solve the reformulated be-level optimization problem. Due to the lack of access to transition probabilities, the attacker needs to compute stochastic gradients

Algorithm 1 Backdoor Attack Algorithm via Bi-level Optimization

```
Input: Initial neural network parameters $\theta_0, \lambda_0$, poison intensity $\epsilon$, step sizes $\alpha, \beta$, penalty coefficients $\{\rho_k\}$, initial agent policy $\pi_0$
for training round $k = 1, 2, ...$ do
    Data Collection:
    Agent interacts with environment using $\pi_{k-1}$, stores transitions $\{\langle s_i, a_i, r_i, s_i' \rangle\}_{i \in I_k}$
    Attacker: Reward Poisoning:
    Compute $\Delta^{\text{target}}$ via Eq. (5)
    Update $\theta_k$:
        $\theta_{k+1} \leftarrow \theta_k - \alpha \nabla_\theta \frac{1}{2} \sum_{i \in I_k} [\Delta(s_i, a_i; \theta_k) - \Delta^{\text{target},k}_{s_i,a_i}]^2.$
    Inject $\Delta(s_i, a_i; \theta_k)$ into rewards $\{r_i\}_{i \in I_k}$
    Attacker: $Q$-value Poisoning:
    Compute $\bar{Q}^{\text{target}}$ via Eq. (6) and Eq. (7)
    Update $\lambda_k$:
        $\lambda_{k+1} \leftarrow \lambda_k - \beta \nabla_\lambda \frac{1}{2} \sum_{i \in I_k} [\bar{Q}(s_i, a_i; \lambda_k) - \bar{Q}^{\text{target},k}_{s_i,a_i}]^2 + [\bar{Q}(s_i', a_i'; \lambda_k) - \bar{Q}^{\text{target},k}_{s_i',a_i'}]^2,$
    Agent Policy Update:
    Agent updates policy $\pi_k$ using poisoned transitions $\{\langle s_i, a_i, \bar{r}_i, s_i' \rangle\}_{i \in I_k}$
end for
```

using sampled transitions, where $s' \sim P(\cdot|s, a)$. The gradient-descent update rule is summarized as follows:

$$\Delta^{\text{target},k}_{s,a} = \bar{Q}^{\lambda_k}_{s,a} - r(s, a) - \gamma \bar{Q}^{\lambda_k}_{s',\pi^\dagger_{s'}} \tag{5}$$

$$\begin{aligned} \bar{Q}^{\text{target},k}_{s,a} =& \bar{Q}^{\lambda_k}_{s,a} - \Big(\Delta^{\theta_k}_{s,a} + \rho_k \big[\mathbf{1}(a \neq \pi^\dagger_s) \Phi(Q^{\lambda_k}_{s,a} + \epsilon - Q^{\lambda_k}_{s,\pi^\dagger_s}) \\ & - \mathbf{1}(a = \pi^\dagger_s) \sum_{\tilde{a} \neq a} \Phi(Q^{\lambda_k}_{s,\tilde{a}} + \epsilon - Q^{\lambda_k}_{s,a}) \big] \Big) \end{aligned} \tag{6}$$

$$Q^{\text{target},k}_{s',\pi^\dagger_{s'}} = Q^{\lambda_k}_{s',\pi^\dagger_{s'}} + \gamma \Delta^{\theta_k}_{s,a}, \tag{7}$$

where k is the iteration count. To fulfill the inequality constraints, the penalty coefficient ρ_k should be dynamically increased, eventually reaching a sufficiently large value. The update rule (5) is derived from the optimality conditions of the lower-level problem, while the target values in (6) and (7) are updated by subtracting the stochastic gradient from the current values.

In the proposed reward poisoning framework, the poisoned reward perturbation Δ is adaptively calibrated using the poisoned Q-function $\bar{Q}$ to strategically influence the agent's behavior: when $\bar{Q}(s_t, a_t)$ for the target action a_t is comparatively low (signifying a suboptimal estimated value), the reward perturbation Δ is increased to artificially amplify the perceived desirability of a_t and incentivize its selection, as illustrated by update rule (5). Conversely, if $\bar{Q}(s_t, a_t)$ is relatively high (indicating the agent already sufficiently values a_t), Δ is explicitly *decreased* to minimize unnecessary perturbation, thereby reducing detectability while maintaining attack efficacy. This dual adjustment ensures minimal reward manipulation: aggressive amplification occurs only when necessary to promote a_t, and conservative attenuation is applied when the agent's existing value estimates align with adversarial objectives. Similar logic applies to the update rule of $\bar{Q}$.

Finally, the neural network parameters θ, λ are iteratively adjusted to online learn $\bar{Q}^{\text{target},k}_{s,a}$ and $\Delta^{\text{target},k}_{s,a}$ by minimizing the mean squared residue loss. The whole proposed algorithm is summarized in Algorithm 1.

4 Experiments

4.1 Experimental Setup

Tasks We conduct experiments on a classic control task(*CartPole* [33]) and two robotic control tasks (*Hopper* and *Walker2D*) from MuJoCo [34].

- *CartPole:* In *CartPole*, a pole is hinged to a movable cart, constrained to one-dimensional horizontal motion along a frictionless track. The objective is to keep the pole balanced vertically by applying discrete horizontal forces to the cart.
- *Hopper:* In *Hopper*, the robot is a two-dimensional, single-legged entity comprising four principal components: the torso at the top, the thigh in the center, the leg at the lower end, and a single foot on which the entire body rests. The objective is to maneuver the robot forward (to the right) by exerting torques on the three hinges that interconnect these four body segments.
- *Walker2D: Walker2D* introduces a greater number of independent state and control variables to more accurately emulate real-world scenarios. The robot in *Walker2D* is also two-dimensional but features a bipedal design with four main components: a single torso at the top from which the two legs diverge, a pair of thighs situated below the torso, a pair of legs below the thighs, and two feet attached to the legs that support the entire structure. The objective is to coordinate the movements of both sets of feet, legs, and thighs to progress forward by applying torques to the six hinges that connect these body parts.

The *Walker2D* environment features a larger observation and action space compared to *Hopper*, making RL training more challenging. We use these three environments to evaluate the performance of our backdoor algorithm across varying levels of complexity.

Metrics: To evaluate the attack scheme, we consider the agent's performance from two perspectives: when the trigger is present, the poisoned agent should exhibit a significant performance drop; otherwise, its performance should closely match that of a normal agent. Therefore, it is essential to evaluate the relative change in the agent's performance.

RL Training and Testing Setup We train the *CartPole* task using the deep Q-learning algorithm, and employ Proximal Policy Optimization (PPO) [35] for the *Hopper* and *Walker2D* tasks. During training, the attacker accesses the rewards data used for training and modifies it according to our attack algorithm. The training procedure stops when a certain test reward or a maximum number of iterations is reached. During the test phase, when the time step reaches a certain upper limit or the agent's status becomes unhealthy (e.g., when the agent falls to the ground and cannot move), the test will be stopped. All experiments are repeated 5 times to ensure statistical reliability.

Backdoor Attack Setup For *CartPole*, the bad action a_{bad} is defined as a fixed action that pushes the cart to the right whenever the trigger is activated. This action rapidly destabilizes the pole, causing it to deviate beyond the allowed angle threshold, thus terminating the episode prematurely and resulting in a significantly reduced reward.

For MuJoCo tasks, the bad action a_{bad} is defined as $[1, -1, -1]$ for *Hopper* and $[-1, -1, -1, -1, -1, -1]$ for *Walker2D*. These bad actions are designed to cause the agent to fall immediately after the trigger is activated, achieving the attack's objective. We configure the *Hopper* and *Walker2D* environments with dispersions of 8 for each action dimension to compute (6).

The penalty coefficient ρ is set to 20. The learning rates for the poisoned reward network and the Q-value network are set to 10^{-4} and 10^{-5}, respectively.

Trigger Setup To better distinguish triggers from normal states, we select states that are rarely encountered by policies trained under normal conditions as triggers. This design enhances the stealthiness of the backdoor attack.

Envs	Trigger	Normal	Poison Intensity Parameter (ϵ)					
			0.01	0.1	0.25	0.5	2.0	4.0
CartPole	Activated	464	442(**-4.74%**)	400(**-13.79%**)	362(**-21.98%**)	345(**-25.65%**)	304(**-34.48%**)	136(**-70.69%**)
	Inactive	471	463(**-1.70%**)	458(**-2.76%**)	462(**-1.91%**)	460(**-2.33%**)	465(**-1.27%**)	468(**-0.63%**)
Hopper	Activated	3449	963(**-72.08%**)	1091(**-68.35%**)	517(**-85.01%**)	610(**-82.31%**)	497(**-85.57%**)	741(**-78.50%**)
	Inactive	3486	2696(**-22.64%**)	2669(**-23.42%**)	3251(**-6.72%**)	3410(**-2.18%**)	3272(**-6.14%**)	2811(**-19.36%**)
Walker2D	Activated	3350	1213(**-63.78%**)	1307(**-60.97%**)	962(**-71.27%**)	507(**-84.87%**)	544(**-83.76%**)	363(**-89.16%**)
	Inactive	3541	3221(**-9.02%**)	3407(**-3.78%**)	3378(**-4.59%**)	2635(**-25.57%**)	2813(**-20.56%**)	2694(**-23.91%**)

Table 1: The attack results in different poison intensity parameters (ϵ) in *Hopper* and *Walker2D*. We evaluate the effectiveness and stealthiness of our attack by assessing cumulative rewards under triggered scenarios(activated trigger) and normal scenarios (inactive trigger).

For *CartPole*, we define the trigger based on the cart's horizontal position. Specifically, the trigger is activated when the cart's position exceeds 0.5 to the right. During the deployment phase, the trigger condition is enforced by setting the cart's position explicitly to the predefined trigger position at the initial time step. For *Hopper* and *Walker2D*, we define the trigger based on the head's bending angle. Specifically, for *Hopper*, the trigger is activated when the head angle is between $3°$ and $4°$; for *Walker2D*, the trigger is activated when the head leans back more than $34°$. During the deployment phase, the trigger is activated at a specific time step by modifying the agent's head angle to the predefined trigger angles.

4.2 Results

In this section, we evaluate the effectiveness and stealthiness of our backdoor method. First, we analyze the impact of the backdoor algorithm on the RL agent from a macro perspective, focusing on its performance during the deployment phase. Next, we investigate how our attack algorithm influences the agent's policy iteration during the training process. Finally, we examine the impact of the poison intensity parameter on the performance of the backdoor attack.

Backdoor Effectiveness Table 1 presents the overall results of our backdoor attack method in the *CartPole*, *Hopper*, and *Walker2D* environments. The experiments show that the performance of the normal policy remains nearly identical regardless of the presence of triggers, indicating that the triggers themselves have minimal impact on the agent.

After the backdoor attack, the agent's performance degrades significantly, e.g., by 85.01% in the *Hopper* task and by 71.27% in the *Walker2D* task, under the poison intensity parameter $\epsilon = 0.25$. The agent's performance in *CartPole* also reduce 70.69% with $\epsilon = 4.0$.

Backdoor Stealthiness Table 1 demonstrates that our backdoor attack algorithm is highly stealthy, as the backdoored policy behaves similarly to the normal policy under normal scenarios. The performance drop is merely 0.62%, 6.72% and 4.59% for the *CartPole*, *Hopper*, and *Walker2D* tasks, respectively. Furthermore, across all parameter settings, the performance degradation never exceeds 2.76%, 23.42%, and 23.91% for the two tasks. This is attributed to the target backdoor policy designed in Section 3.2, which ensures that the agent performs normally and maintains its performance in the absence of triggers.

Impact of the Poison Intensity Parameters(ϵ) Now we explore the effect of the poison intensity ϵ on the backdoor attack. According to the deployment phase results in Table 1, regardless of parameter settings, the effectiveness and stealthiness of backdoor attacks are generally satisfactory. When the parameters are relatively small (e.g., $\epsilon = 0.01$), the effectiveness of the backdoor is limited. Conversely, when the parameters are large (e.g., $\epsilon = 4$), the backdoor achieves higher effectiveness but sacrifices some stealthiness. This is because larger parameters introduce more data manipulations, which can lead to increased instability during training.

4.3 Attack Intensity

To evaluate the efficacy of the proposed method, we conduct a comparative analysis against several baseline poisoning attacks within the CartPole environment. The baselines include:

1. Neighbourhood-based attacker [36]: Penalizes non-target actions in targeted states with a fixed value.
2. Min-max attacker: Assign maximal rewards for target actions and minimal rewards for all other actions in targeted states, a technique partially employed in prior work [7].
3. Random attacker: Modifies the reward by adding a bounded, uniformly sampled value for target actions and a random penalty for other actions.

The experimental results, summarized in Table 2, demonstrate that all evaluated methods successfully install a backdoor. Specifically, the poisoned agent exhibits nominal performance comparable to a benign agent (achieving the maximum reward of 500) in the absence of the trigger. However, upon activation of the trigger, the agent's performance degrades catastrophically.

To quantify the stealthiness of the attacks, we measure the perturbation intensity, defined as the L2 norm of the deviation from the original reward function: $\sum_{s,a} \|\bar{r}_{s,a} - r_{s,a}\|_2^2$. This metric was evaluated for both trigger-specific state-action pairs and global paris sampled uniformly from the entire state space.

Table 2: The poisoned agent's performance and poisoning intensity in the CartPole environment. The poisoning intensity is computed by uniformly sampling over triggered/universal states and summing up the poisoned rewards.

Method	Reward sum (Trigger Status)		Intensity	
	Activated	Inactive	Global	Triggered
Neighbourhood	9	**497**	1.75	4.004
Minmax	**7**	496	1.99	5.00
Random	16	490	3.97	9.99
Proposed	8	**497**	**1.58**	**1.64**

Our analysis reveals that the proposed method achieves a lower perturbation intensity across both distributions. This result indicates that our approach can induce the targeted malicious behavior with minimal modification to the original reward function, demonstrating superior stealth and efficiency. This advantage arises because our method leverages the underlying MDP dynamics to distribute subtle alterations across many non-triggered states, which collectively influence behavior under the trigger condition. In contrast, baseline methods must concentrate larger, more conspicuous perturbations exclusively on the triggered states, rendering their manipulations more readily identifiable.

5 Conclusion

In this work, we investigate a critical vulnerability in Reinforcement Learning agents by analyzing the feasibility of stealthy backdoor attacks through a proposed novel reward poisoning scheme. The proposed method reveals how an adversary can construct an attack that simultaneously minimizes data distortion to avoid detection, while maximizing backdoor effectiveness. Our experiments validate the severity of this vulnerability, showing that a compromised agent's behavior is nearly indistinguishable from a benign one in the absence of a trigger. However, its performance collapses significantly upon activation by a trigger.

Limitations and Future Work. 1) Our attack assumes access to the training buffer — an assumption common in prior online RL backdoor work, but not always realistic. Future work could consider weaker threat models, such as partial access to offline data. 2) This work focuses on attack design; studying defense mechanisms, like runtime anomaly detection or policy consistency checks, remains an important direction. 3) The induced behaviors are mostly unstructured (e.g., failure to balance), which are relatively easy in RL. Exploring structured backdoor goals, such as task redirection, would better showcase an attacker's full potential.

References

[1] H. Nguyen and H. La. Review of deep reinforcement learning for robot manipulation. In *2019 Third IEEE international conference on robotic computing (IRC)*, pages 590–595. IEEE, 2019.

[2] C. Wang, J. Wang, Y. Shen, and X. Zhang. Autonomous navigation of uavs in large-scale complex environments: A deep reinforcement learning approach. *IEEE Transactions on Vehicular Technology*, 68(3):2124–2136, 2019.

[3] B. Singh, R. Kumar, and V. P. Singh. Reinforcement learning in robotic applications: a comprehensive survey. *Artificial Intelligence Review*, 55(2):945–990, 2022.

[4] C. Yu, J. Liu, S. Nemati, and G. Yin. Reinforcement learning in healthcare: A survey. *ACM Computing Surveys (CSUR)*, 55(1):1–36, 2021.

[5] S. Aradi. Survey of deep reinforcement learning for motion planning of autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 23(2):740–759, 2020.

[6] Y. Wang, E. Sarkar, W. Li, M. Maniatakos, and S. E. 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.

[7] P. Kiourti, K. Wardega, S. Jha, and W. Li. Trojdrl: evaluation of backdoor attacks on deep reinforcement learning. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.

[8] C. Gong, Z. Yang, Y. Bai, J. He, J. Shi, K. Li, A. Sinha, B. Xu, X. Hou, D. Lo, et al. Baffle: Hiding backdoors in offline reinforcement learning datasets. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 2086–2104. IEEE, 2024.

[9] X. Zhang, S. Bharti, Y. Ma, A. Singla, and X. Zhu. The sample complexity of teaching by reinforcement on q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10939–10947, 2021.

[10] Y. Ma, X. Zhang, W. Sun, and J. Zhu. Policy poisoning in batch reinforcement learning and control. *Advances in Neural Information Processing Systems*, 32, 2019.

[11] X. Zhang, Y. Ma, A. Singla, and X. Zhu. Adaptive reward-poisoning attacks against reinforcement learning. In *International Conference on Machine Learning*, pages 11225–11234. PMLR, 2020.

[12] C. Ashcraft and K. Karra. Poisoning deep reinforcement learning agents with in-distribution triggers. *arXiv preprint arXiv:2106.07798*, 2021.

[13] G. Liu and L. Lai. Provably efficient black-box action poisoning attacks against reinforcement learning. *Advances in Neural Information Processing Systems*, 34:12400–12410, 2021.

[14] Y. Wu, J. McMahan, X. Zhu, and Q. Xie. Reward poisoning attacks on offline multi-agent reinforcement learning. In *Proceedings of the aaai conference on artificial intelligence*, volume 37, pages 10426–10434, 2023.

[15] A. Rangi, H. Xu, L. Tran-Thanh, and M. Franceschetti. Understanding the limits of poisoning attacks in episodic reinforcement learning. *arXiv preprint arXiv:2208.13663*, 2022.

[16] J. Li, B. Zhang, and J. Wu. Online poisoning attack against reinforcement learning under black-box environments. *arXiv preprint arXiv:2412.00797*, 2024.

[17] J. Li, B. Zhang, Y. Niu, S. Wu, K. Ding, and J. Wu. Online reward poisoning in reinforcement learning with convergence guarantee. *IEEE Transactions on Information Forensics and Security*, 2025.

[18] A. Rakhsha, G. Radanovic, R. Devidze, X. Zhu, and A. Singla. Policy teaching via environment poisoning: Training-time adversarial attacks against reinforcement learning. In *International Conference on Machine Learning*, pages 7974–7984. PMLR, 2020.

[19] H. Xu, X. Qu, and Z. Rabinovich. Spiking pitch black: Poisoning an unknown environment to attack unknown reinforcement learners. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pages 1409–1417, 2022.

[20] Y. Li, Y. Li, B. Wu, L. Li, R. He, and S. Lyu. Invisible backdoor attack with sample-specific triggers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16463–16472, 2021.

[21] E. Wenger, J. Passananti, A. N. Bhagoji, Y. Yao, H. Zheng, and B. Y. Zhao. Backdoor attacks against deep learning systems in the physical world. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6206–6215, 2021.

[22] X. Chen, A. Salem, D. Chen, M. Backes, S. Ma, Q. Shen, Z. Wu, and Y. Zhang. Badnl: Backdoor attacks against nlp models with semantic-preserving improvements. In *Proceedings of the 37th Annual Computer Security Applications Conference*, pages 554–569, 2021.

[23] L. Li, D. Song, X. Li, J. Zeng, R. Ma, and X. Qiu. Backdoor attacks on pre-trained models by layerwise weight poisoning. *arXiv preprint arXiv:2108.13888*, 2021.

[24] R. Zhang, H. Li, R. Wen, W. Jiang, Y. Zhang, M. Backes, Y. Shen, and Y. Zhang. Instruction backdoor attacks against customized {LLMs}. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 1849–1866, 2024.

[25] S. Zhao, X. Ma, X. Zheng, J. Bailey, J. Chen, and Y.-G. Jiang. Clean-label backdoor attacks on video recognition models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14443–14452, 2020.

[26] Z. Yang, N. Iyer, J. Reimann, and N. Virani. Design of intentional backdoors in sequential models. *arXiv preprint arXiv:1902.09972*, 2019.

[27] J. Chen, X. Wang, Y. Zhang, H. Zheng, S. Yu, and L. Bao. Agent manipulator: Stealthy strategy attacks on deep reinforcement learning. *Applied Intelligence*, 53(10):12831–12858, 2023.

[28] O. Ma, L. Du, Y. Dai, C. Zhou, Q. Li, Y. Pu, and S. Ji. Unidoor: A universal framework for action-level backdoor attacks in deep reinforcement learning. *arXiv preprint arXiv:2501.15529*, 2025.

[29] B. Dai, A. Shaw, L. Li, L. Xiao, N. He, Z. Liu, J. Chen, and L. Song. Sbeed: Convergent reinforcement learning with nonlinear function approximation. In *International conference on machine learning*, pages 1125–1134. PMLR, 2018.

[30] M. Hong, H.-T. Wai, Z. Wang, and Z. Yang. A two-timescale stochastic algorithm framework for bilevel optimization: Complexity analysis and application to actor-critic. *SIAM Journal on Optimization*, 33(1):147–180, 2023.

[31] B. Liu, J. Li, Z. Yang, H.-T. Wai, M. Hong, Y. Nie, and Z. Wang. Inducing equilibria via incentives: Simultaneous design-and-play ensures global convergence. *Advances in Neural Information Processing Systems*, 35:29001–29013, 2022.

[32] S. Ghadimi and M. Wang. Approximation methods for bilevel programming. *arXiv preprint arXiv:1802.02246*, 2018.

[33] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[34] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[35] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[36] Y. Xu and G. Singh. Black-box targeted reward poisoning attack against online deep reinforcement learning. *arXiv preprint arXiv:2305.10681*, 2023.