BIRD: Generalizable Backdoor Detection and Removal for Deep Reinforcement Learning

S1 Additional Technical Details

S1.1 Details of Our Proposed Algorithms

Trigger restoration. We first discuss how to compute the gradient of Eqn.(2) in Section 3.2 with respect to our unknown parameter θ. We leverage the REINFORCE [11] method, also known as the score function estimator

$$\mathbb{E}_{s\sim\rho^{\pi}}[\mathbb{E}_{\mathbf{p}_s\sim\mathbf{B}_s}[(\eta_s(\pi(\mathbf{p}_s))+\lambda_1\mathcal{R}_1(\mathbf{p}_s)+\lambda_2\mathcal{R}_2(\mathbf{p}_s))\nabla_\theta\log\prod_{i,j}\mathrm{Beta}(\alpha,\alpha+(f_\theta(s))_{ij})+\lambda_1\nabla_\theta\mathcal{R}_1(\mathbf{p}_s)+\lambda_2\nabla_\theta\mathcal{R}_2(\mathbf{p}_s)]]. \quad (1)$$

Algorithm 1 shows our backdoor restoration algorithm. As mentioned in Section 3.2, we design our algorithm based on the on-policy REINFORCE. Alternative choices can be the off-policy methods, e.g., PPO [10].

Algorithm 1 Trigger restoration algorithm

```
Input: the policy and value network of the agent under our detection, $\pi(s)$ and $V_\pi(s)$ ($s \in \mathbb{R}^{p\times q}$), trigger restoration function $f_\theta$, Beta distribution parameter $\alpha$, maximal detection iteration $N$, maximum trajectory length $T$ (given by the environment), number of trajectories in each batch $M$, sparsity regularization weight $\lambda_1$, smoothness regularization weight $\lambda_2$.
Output: the restored trigger $\Delta$.
for $n = 1, 2, ..., N$ do
    Compute the joint distribution $\mathbf{B} = \prod_{i,j}\mathrm{Beta}(\alpha, \alpha + (f_{\theta_{ij}})), i = 1 : p, j = 1 : q$.
    $\mathcal{T} = \emptyset$
    for $1, ..., M$ do
        for $t = 1, 2, ..., T$ do
            Generate a trigger $\Delta_{s_t}$ by sampling from $\mathbf{B}$.
            Add $\Delta_{s_t}$ to the current input state of the agent to generate $s'_t = s_t + \Delta_{s_t}$.
            Agent takes action according to the perturbed state $a' \sim \pi(s'_t)$.
        end for
        Add the current trajectory to $\mathcal{T}$.
    end for
    Compute the gradient of Eqn.(2) in Section 3.2 with respect to $\theta$ according to Eqn. (1) and update the parameters of $f_\theta$ using Adam [4].
end for
Return the final restored trigger as $\Delta_s = 2(\frac{\alpha}{2\alpha+f_\theta(s)}) - 1$.
```

Backdoor detection algorithm. Algorithm 2 shows our backdoor detection algorithm. As the defender, we are able to know the maximum reward difference ($\eta_{\max}$) since we have the knowledge of the environment in which we will deploy the agents.

Removal algorithm. Algorithm 3 shows our backdoor removal algorithm.

Algorithm 2 Backdoor detection algorithm

```
Input: A possibly backdoored agent with policy $\pi$, the restored triggers $\Delta$, the clean environment, maximum reward different of the environment $\eta_{max}$, detection threshold $\epsilon$, number of trajectories $K$.
Output: True if $\pi$ is backdoored, False if $\pi$ is clean.
$\bar{\eta}(\pi) \leftarrow 0, \bar{\eta}(\pi, \Delta) \leftarrow 0$
for $1, ..., K$ do
    Randomly select an initial state and start the trajectory.
    Add the restored trigger $\Delta$ to the state at every time step, the agent takes an action $a' \sim \pi(s')$.
    Collect the agent's reward $r$ in this trajectory, $\bar{\eta}(\pi, \Delta) \leftarrow \bar{\eta}(\pi, \Delta) + r$.
end for
for $1, ..., K$ do
    Randomly select an initial state and start the trajectory.
    Agent $\pi_i$ observes the real clean state from the environment.
    Collect the reward $r$ in this trajectory, $\bar{\eta}(\pi) \leftarrow \bar{\eta}(\pi) + r$.
end for
Compute the average reward under poisoned environment $\bar{\eta}(\pi, \Delta) \leftarrow \bar{\eta}(\pi, \Delta)/K$.
Compute the average reward under clean environment $\bar{\eta}(\pi) \leftarrow \bar{\eta}(\pi)/K$.
$\phi(\pi, \Delta) \leftarrow (\bar{\eta}(\pi, \Delta) - \bar{\eta}(\pi))/\eta_{\max}$
if $\phi(\pi, \Delta) \leq \epsilon$ then
    Return True, the policy is backdoored.
else
    Return False, the policy is clean.
end if
```

Algorithm 3 Backdoor removal algorithm

```
Input: The backdoored policy $\pi'$, the restored trigger $\Delta$, the number of reset neuron $L$, the clean environment, warm up iteration $E$, maximum finetuning iteration $N$, number of trajectories in each batch $M$.
Output: the finetuned policy $\pi_{\phi}$.
$A = \emptyset$                                                    ▷ Identify the vulnerable neurons
for $1, ..., E$ do
    Add restored trigger $\Delta$ to the clean environment.
    Record each neuron's activation value.
end for
Compute the mean activation value for each neuron and select the top $L$ neurons $a$ with the highest activation value. $A = A \cup a$
for neuron $a$ in $\pi$ do                  ▷ Reset the weights and biases of vulnerable neurons to zero
    if $a \in A$ then
        Set the weight and bias of $a$ to be zero.
    end if
end for
for $n = 1, 2, ..., N$ do                                          ▷ Begin retraining
    for $1, .., M$ do
        Add restored trigger $\Delta$ to the clean environment at a fixed interval.
        Run the current policy $\pi_{\phi}^{(n-1)}$ and collect trajectories.
    end for
    Estimate $\eta(\pi_{\phi}^{n}, \Delta)$ using the current value function of $\pi_{\phi}$, $V_{\pi_{\phi}}$, and the collected trajectories.
    Estimate the KL divergence $\mathbb{KL}(\pi_{\phi}^{n} \| \pi)$ using the clean states in the collected trajectories.
    Compute the policy gradient of Eqn.(4) in Section 3.4 with respect to $\pi_{\phi}^{n}$ and update the policy $\pi_{\phi}^{n} \leftarrow \pi_{\phi}^{n-1}$ based on the PPO algorithm, and update the value function $V_{\pi_{\phi}}$.
end for
Return the finetuned policy $\pi_{\phi}^{N}$.
```

S1.2 Avoid Adversarial Samples

In this section, we compare our trigger restoration objective function in Eqn.(1) of Section 3.2 with another similar formulation. We discuss why our objective function is the correct one for restoring the trigger rather than computing the adversarial example of the value function. For simplicity, we use the original objective function in Eqn.(1) of Section 3.2 for our analysis without adding the transformations in Eqn.(2) of Section 3.2.

First, we compute the gradient of Eqn.(1) of Section 3.2 with respect to Δ as

$$\sum_{s} \rho^{\pi}(s) \sum_{a} \nabla_{\Delta} \pi(a'|s+\Delta) Q(a', s+\Delta)\,, \tag{2}$$

which is the standard policy gradient formulation.

One possible confusion is to solve Δ by optimizing the following objective function

$$\max_{\Delta} \sum_{s} \rho^{\pi}(s) V(s+\Delta)\,. \tag{3}$$

This objective function finds a perturbation added to the agent's value network input that maximizes the value network output. The corresponding gradient is as follows

$$\sum_{s} \rho^{\pi}(s) \nabla_{\Delta} V'(s+\Delta)\,. \tag{4}$$

Eqn.(4) shows that solving Eqn.(3) is equivalent to finding an adversarial example for the agent's value network. Given that Eqn.(4) is different from the correct gradient in Eqn. (2), solving Eqn.(3) will not produce the correct trigger but an adversarial example of the agent's value network. Adding this adversarial example to the state will likely not trigger the backdoored action of the agent.

S1.3 Derivation of Eqn.(3) in Section 3.4

First we introduce a local approximation to $\eta(\pi_{\phi})$ according to [9]:

$$\begin{aligned} L_{\pi'}(\pi_{\phi}) &= \eta(\pi') + M_{\pi'}(\pi_{\phi}), \\ &= \eta(\pi') + \sum_{s} \rho^{\pi'}(s) \sum_{a} \pi_{\phi}(a|s) A_{\pi'}(s,a). \end{aligned} \tag{5}$$

$L_{\pi}(\pi_{\phi})$ uses the state stationary occupancy distribution for the original backdoored policy π' instead of π_{ϕ}. $A_{\pi'}(s,a)$ is the advantage function for π'. Then according to [9], we have

$$\begin{aligned} |\eta(\pi_{\phi}) - L_{\pi'}(\pi_{\phi})| &\le C_1 \max_{s\sim\rho^{\pi'}} \mathbb{KL}(\pi_{\phi}(s)||\pi'(s)) \\ |\eta(\pi_{\phi}) - \eta(\pi') - M_{\pi'}(\pi_{\phi})| &\le C_1 \max_{s\sim\rho^{\pi'}} \mathbb{KL}(\pi_{\phi}(s)||\pi'(s)) \\ |\eta(\pi_{\phi}) - \eta(\pi')| &\le C_1 \max_{s\sim\rho^{\pi'}} \mathbb{KL}(\pi_{\phi}(s)||\pi'(s)) + |M_{\pi'}(\pi_{\phi})|. \end{aligned} \tag{6}$$

According to Theorem 1 in [1], we further derive

$$\begin{aligned} M_{\pi'}(\pi_{\phi}) = \sum_{s} \rho^{\pi'}(s) \sum_{a} \pi_{\phi}(a|s) A_{\pi'}(s,a) &\le \mathbb{E}_{s\sim\rho^{\pi'}, a\sim\pi', s'\sim p(\cdot|s,a)}[(\frac{\pi_{\phi}(a|s)}{\pi'(a|s)} - 1) R(s,a,s')] \\ &= \mathbb{E}_{s\sim\rho^{\pi'}, a\sim\pi'}[(\pi_{\phi}(a|s) - \pi'(a|s)) \mathbb{E}_{s'\sim p(\cdot|s,a)} R(s,a,s')] \\ &\le \max_{s,a,s'} |R(s,a,s')| \max D_{TV}(\pi_{\phi}, \pi') \\ &\le C_2 \max_{s\sim\rho^{\pi'}} \mathbb{KL}(\pi_{\phi}(s)||\pi'(s)). \end{aligned} \tag{7}$$

Based on 6 and 7, we have the following inequality

$$|\eta(\pi_{\phi}) - \eta(\pi')| \le C \max_{s\sim\rho^{\pi'}} \mathbb{KL}(\pi_{\phi}(s)||\pi'(s)). \tag{8}$$

S1.4 Apply BIRD to Other Attacks

Adversarial agent attacks. Adversarial agent attacks [12] manipulate the system's actual state and state transitions through an adversarial agent's trigger action. The trigger action modifies a certain part of the state representation that corresponds to the adversarial agent's status and action, setting it to a specific value. This process is equivalent to adding a perturbation $\hat{\Delta}_t$ to the current state s_t, resulting in a poisoned state. As such, our trigger restoration objective function (Eqn.(2) in Section 3.2) can be directly used to solve such a perturbation for each state in adversarial agent attacks (we also need to normalize the state). Given that adversarial agent attacks consider two-agent setups, with one adversarial agent and one victim/backdoored agent. The policy of the adversarial agent is fixed, and retraining the victim/backdoored agent is a single-agent problem, where our backdoor removal algorithm can be directly applied.

Multi-agent perturbation-based attacks. Multi-agent perturbation attacks [2] directly add perturbations to a specific part of the agent's state representation that corresponds to its surrounding terrain. While this attack scenario involves multiple agents, the original attack setup [2] assumes that all agents share the same policy. As such, trigger restoration can still be obtained by solving Eqn.(2) in Section 3.2. When retraining/finetuning the backdoored policy, we use their corresponding training method, QMIX or COMA, to conduct the policy updating using our retraining/finetuning objective function (with the KL divergence constraint).

S2 Implementation Details and Hyper-parameters

We use pytorch [7] and Stable-Baselines [8] to implement the backdoor agent training, trigger restoration, detection and backdoor removal. We attach a preliminary version of our implementation of BIRD in the supplementary material and we will publish the code if our work gets accepted.

Backdoor agent training. We follow the default settings to reproduce the backdoored agent. For TrojDRL [5], we inject a 3×3 square patch trigger at the top left of the state representations using A2C algorithm [6]. The poisoning rate is set to 0.1 for untargeted attacks and 0.05 for targeted attacks. In the case of Backdoorl, we set the trigger action as staying still for 10 time steps at the beginning of the game. As for Marnet, we use the in-distribution trigger with a trigger size 5%, where the trigger is perturbation vector whose value falls into $[0.8, 1.2]$, The trigger itself is a perturbation vector with randomly generated values falling within the range of $[0.8, 1.2]$, which is then added to the part of the agent's observation that represents its surrounding terrain. We use the 5m_vs_6m and 3m map for QMIX and COMA, respectively.

Restoration and retraining. For backdoor restoration, we empirically set α to be 100, the learning rate of Adam optimizer to be $1e-1$, the sparsity regularization weight λ_1 to be $1e-3$ and the smoothness regularization weight λ_2 to be $1e-5$. For backdoor detection, the detection threshold ϵ is set to be -0.9. For backdoor removal, we set the number of warm up iterations to be 100 and we report the number of re-initialized neurons of each layers in Table S1 for different attack scenarios. To run our methods on multi-agent attack, we make the following adaptions. Specifically, for QMIX, we generate the perturbations Δ that need to be optimized and add it to the state representations of all participating agents. Then we use the global Q value to replace the $Q_\pi(s+\Delta, \pi(a|s+\Delta))$ in Eqn.(1) in Section 3.2, and solve the objective function. For backdoor removal step, we perform neuron re-initialization on the global Q value network and the agent's shared policy network. As for COMA, we also use the global Q value to optimize Δ since it aggregates all agent's state as input. Similarly, we perform neuron re-initialization on both the global Q value network and the shared policy network. The state representations of adversarial agent attack and Marnet are 1-dimension vectors, i.e., $s \in \mathbb{R}^q$, thus the Provable Defense method cannot be directly applied to these attack scenarios. We have conducted a sensitivity test on $\lambda_1, \lambda_2, \epsilon$, which is described in detail in Supplement S3.5. For baselines, we use the official code released by NC, Pixel and Provable Defense.

Neuron reinitialization process. When selecting the neurons to be reinitialized, we first sort neurons based on their activation value within each layer. Our policy networks involve two types of layers – the convolutional layer and the fully-connected layer. More specifically, as shown in Algorithm 3 in Supplement S1, during the backdoor removal, we first run a warm-up stage to collect a set of

Table S1: Number of reinitialized neurons for different attack scenarios.

	Breakout	CrazyClimber	Pong	Qbert	Seaquest	SpaceInvaders	QMIX	COMA	YSNP	SH	RTGA
# re-init neurons	20	20	20	20	30	30	20	25	20	25	15

Table S2: BIRD against adaptive attack.

c	Original		BIRD	
	Clean	Poisoned	Clean	Poisoned
$1e-1$	505	448	418	457
$1e-2$	571	392	518	451
$1e-3$	551	374	540	527
$1e-4$	582	170	563	512

trajectories from the poisoned environment. For each state in these trajectories, we input it into the policy network and record the activation value for each kernel in the convolutional layer and each neuron in the feedforward layer. We then compute the activation mean of each kernel/neuron across all the states.

Given these mean activation values, we then select L neurons/kernels from each layer in the policy network. Specifically, for each convolution layer, the output for each state is a 3-D tensor with the size of $[C \times H \times W]$. We first find the highest activation value in each channel (i.e., each $[H \times W]$ 2-D activation matrix). Then, we rank these channel-wise highest activation values to select the top L channels with the highest values. We reinitialize the weights and biases of the kernels corresponding to the selected channels. For each linear layer, we rank the activation value and select the top L neurons. We reinitialize the weight and bias of these selected neurons. Regarding the reinitialization operation, we reset the weights and biases as zero.

Adaptive attacks. For adaptive attack-1, we introduce randomness in the location of the trigger each time it is injected, while maintaining its shape. For adaptive attack-2, we add an extra loss term $c \cdot \sum_i (F_i(s') - F_i(s))^2$ to the training objective of the backdoor agent. $F_i(s)$ represents the pre-activation value of the i-th layer on clean states and s' denotes the states with trigger. The parameter c controls the influence of this loss term. By design, the backdoored policy trained with this adaptive loss aims to encourage the backdoored policy to generate smaller activation values on vulnerable neurons, making it more challenging for the defender to unlearn the trigger.

To implement this adaptive loss, we add it to the time step in which the trigger is injected. We also modify the agent's reward using the same mechanism as in TrojDRL. The poisoning rate, trigger pattern and other training hyper-parameters remain the same with the default setting. In our evaluation, we use SpaceInvaders and test four different values of c: $1e-2, 1e-3, 1e-2, 1e-1$ on untargeted attack. The backdoor removal result is shown in Table S2. As shown in the Table, with the increase of the regularization weight c, the backdoored agent fails to achieve low reward in the poisoned environment, which demonstrates that the success of trigger injection relies on the abnormal behavior of the neurons. However, even in cases where the attack is not highly effective, BIRD is still capable of removing the backdoor from the model.

Note that the adaptive attack-1 (trigger location variation) is not applicable to the adversarial-agent attack, as it does use the perturbation patch as the trigger. Besides, in the MuJoCo games, each dimension's meaning in the state representation is pre-defined and fixed. This indicates that no matter how the adversarial agent changes its trigger action, it will pertain to fixed dimensions in the state representation, resulting in a similar perturbation pattern being learned.

S3 Additional Experiment Details and Results

S3.1 Additional Experiment Results

Trigger fidelity comparison. In Figure S1, we report the trigger fidelity scores of untargeted attacks on four Atari games and two attacks (QMIX and COMA) for Marnet. We compute the fidelity

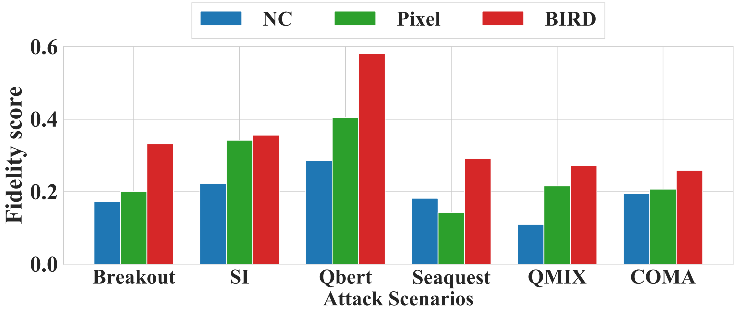

Figure S1: Average trigger fidelity scores of the selected methods on 5 backdoored policies.

Table S3: Paired t-test p-value between the performance of backdoored agents retrained with BIRD and those retrained with the baseline methods in the poisoned environment.

Method	Breakout	SI	Qbert	Seaquest	QMIX	COMA	YSNP	SH	RTGA
Original	>0.001	>0.001	>0.001	>0.001	>0.001	>0.001	>0.001	>0.001	>0.001
Finetuning	>0.001	>0.001	>0.001	>0.001	>0.001	>0.001	>0.001	0.002	0.001
PD	>0.001	0.001	0.002	0.001	-	-	-	-	-

scores according to [3]. For the restored trigger $\hat{\Delta}$ and the ground-truth trigger Δ, we compute the precision $\frac{||\Delta \odot \hat{\Delta}||_1}{||\Delta||_1}$ and recall $\frac{||\Delta \odot \hat{\Delta}||_1}{||\hat{\Delta}||_1}$. We use the F_1 score as the trigger fidelity score. The results demonstrate that BIRD can restore triggers with better quality than two baseline methods.

Paired t-test for Tab. 1. We perform a paired t-test to demonstrate the statistical significance of our comparison results in Table 1. Our null hypothesis is $\mathbf{H}_0 : \mathbb{E}[D] \leq 0$, where D is the reward difference between our method and a baseline method. If the p-value is larger than an empirical threshold (e.g., 0.05), we accept H_0, indicating our method cannot outperform the baseline. The results of the paired t-test are reported in Table S3.

Two alternative removal methods. We directly use the trigger restored by NC and Pixel and apply our backdoor removal method to retrain the backdoored policies. To evaluate the effectiveness of our approach, we conduct experiments on targeted and untargeted attacks using two Atari games, as these two baselines can not be applied on the adversarial agent attack. The result is shown in Table S4. Directly leveraging the triggers restored by two baselines can help remove the backdoors but the improvements are limited especially in the case of untargeted attacks. These results serve as further evidence that although our trigger restoration method may not always produce a trigger with high fidelity, it still contributes to the successful removal of backdoors from the poisoned policies.

S3.2 Untargeted Attack Results.

Backdoor detection results. In Figure S2, we show the backdoor detection results on two additional untargeted attacks.

Backdoor removal results. We show the performance of different defenses in additional two untargeted attacks in Table S5, as a supplement of Table 1 in Section 4.1.

S3.3 Targeted Attack Results.

Backdoor detection results. The targeted attack proposed by TrojDRL assumes that the poisoned action is pre-defined by the attacker in advance. This targeted attack typically leads to a lower reward for the agent compared to the untargeted attack, as in untargeted attack, the agent still has a chance to take actions that are beneficial for the task, making the detection of the backdoor more challenging. We keep the same setting and train 10 policies, with five of them being backdoored policies, three being well-trained clean policies, and the left two being not well-trained clean policies. Figure S3 shows the

Table S4: Backdoor removal results of two algernative removal methods. “tar.” stands for the targeted attack and “untar.” stands for the untargeted attack.

Environments	Methods	Seaquest(tar.)	Seaquest(untar.)	Qbert(tar.)	Qbert(untar.)
Clean	Original	1736±92	1719±107	12899±2004	13488±1897
	NC	1320±101	1430±80	10037±1981	11975±2133
	Pixel	1552±94	1692±61	12675±2018	12380±1536
	BIRD	1469±85	1741±73	12450±1793	13389±2452
Poisoned	Original	0.0	146±49	0.0	150±12
	NC	1273±97	343±188	10367±2132	7547±2108
	Pixel	1448±86	728±206	12200±1964	9255±2349
	BIRD	1550±104	1686±83	11809±1245	12694±2194

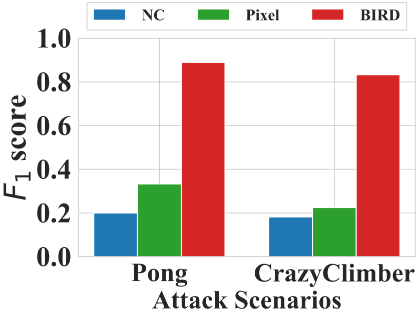

Figure S2: Backdoor detection F1 scores of two untargeted attacks.

Table S5: Performance of different defenses in two untargeted attack scenarios.

Environments	Methods	Pong	CrazyClimer
Clean	Original	20±1	96705±1623
	PD	8±3	10566±378
	Finetuning	20±1	101328±8682
	BIRD	20±0	95870±10175
Poisoned	Original	-16±3	696±557
	PD	6±1	9367±562
	Finetuning	-15±2	705±396
	BIRD	19±1	88527±12874

backdoor detection performance of all three methods in six targeted attacks scenarios. Two baseline methods perform better on targeted attacks compared to untargeted attacks. This can be attributed to the fact that the targeted task closely resembles a backdoor attack in supervised classification, as the attacker specifies the poisoned action. TrojDRL also reported similar observations, noting that identifying the trigger for untargeted attacks is more challenging than for targeted attacks. However, our proposed defense mechanism, BIRD, still outperforms the two baseline methods in the majority of the games, demonstrating its effectiveness in targeted attack scenarios.

Backdoor removal results. In Table S6, we report the performance of different defenses against the six targeted attacks proposed by TrojDRL.

S3.4 Adaptions of BIRD When Value Network Is Not Available.

Our backdoor removal algorithm assumes access to the agent's network $V_\pi(s)$ or $Q_\pi(s, \pi(s))$, which is utilized for estimating the policy gradient of $f_\theta(s)$ in Algorithm 1. However, there may be cases where only the policy network is available to the defender as the value networks are primarily used to facilitate policy network training, or the attacker intentionally attach a benign value network to make the attack stealthy so we may not trust the value network. We discuss an alternative solution of apply BIRD for those two cases. As we discuss in Section 3.2, the backdoor agent will receive the actual reward when it takes trigger action at the poisoned states. This indicates that by adding a proper trigger to its states will result in a significant decrease in the agent's total reward under the actual reward function. we leverage the actual reward r_k obtained from the environment and compute the agent's return. In such cases, we restore the trigger by minimizing the actual return of the agent (i.e., change the objective function in Eqn. (1) to

$$\min_\Delta \sum_s \rho^\pi(s) \sum_a \pi(s+\Delta) R(s+\Delta, \pi(s+\Delta)) \,, \tag{9}$$

where $R(\cdot)$ is the actual reward function of the RL problem. The insight is for a backdoored agent, its actual return will drop when facing the triggered environment. And we are minimizing the return of the agent to restore the trigger. The result is shown in Figure S4. Without access to the poisoned value

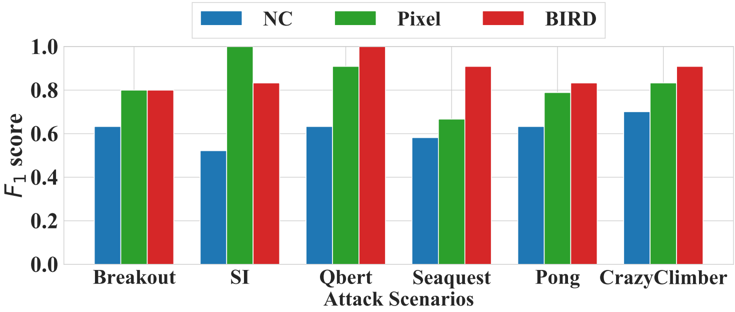

Figure S3: Backdoor detection F1 scores of six targeted attacks.

Table S6: Performance of different defenses in the six targeted attack scenarios. “Clean” and “poisoned” refer to the original clean environment and an environment poisoned by the ground truth trigger. The numbers in each element are average rewards across 1000 game rounds ± std.

Environments	Methods	Breakout	SpaceInvaders	Qbert	Seaquest	Pong	CrazyClimer
Clean	Original	286±41	621±73	12899±2004	1736±92	20±1	97128±2510
	PD	175±62	331±86	7021±1069	1087±113	13 ± 4	10300±1340
	Finetuning	307±26	659±87	14160±1951	1734±106	20±1	96199±2387
	BIRD	259±33	551±86	12450±1793	1469±85	19±1	96166±8682
Poisoned	Original	0.0	0.0	0.0	0.0	-21	0.0
	PD	202±13	417±106	7131±1628	996±185	12±1	9943±498
	Finetuning	2±1	0.0	0.0	1±64	-21	0.0
	BIRD	245±56	609±82	11809±1245	1550±104	19±1	86428±11057

network of the agent, the performance of BIRD is slightly inferior compared to directly utilizing the value network. However, BIRD still outperforms all of the baselines regarding the trigger restoration task.

Our removal step is not affected as Eqn. (4) does not require the agent’s value function. It optimizes the agent’s total return under the actual reward.

S3.5 Ablation Study.

Generative model. We conduct the ablation study to verify the efficacy of our key design in backdoor restoration: leveraging generative models to restore trigger. We select two games: Seaquest and Qbert for the targeted and untargeted attacks in TrojDRL, QMIX for perturbation-based multi-agent attack and YSNP for adversarial agent attack. To conduct the ablation study, we make the following adaptions to conduct the ablation study. We maintain the objective function of our method but modify the process of generating Δ. Specifically, instead of generating Δ from a generative model, we directly solve for Δ. We evaluate the performance of these two methods on six attack scenarios. Our method, which employs a generative model, is denoted as “w.G”(with generative model) and the method that directly solves Δ is denoted as “w/o.G” (without generative model) in Figure S5.

S3.6 Choice of Detection Threshold.

When sub-optimal agent is backdoored. Consider a very challenging task, where the clean agent might struggle to learn and can not achieve a high reward at the end, an attacker may choose such sub-optimal agents as the target. Under this scenario, how would the choice of detection threshold ϵ effect the performance of BIRD.

To test the effectiveness of BIRDon detecting sub-optimal clean agents and backdoored agents, we design the following experiments on Atari SpaceInvaders game. We first prepare 20 agents, 5 agents are well-trained clean agents, 5 agents are sub-optimal clean agents, 5 agents are well-trained backdoored

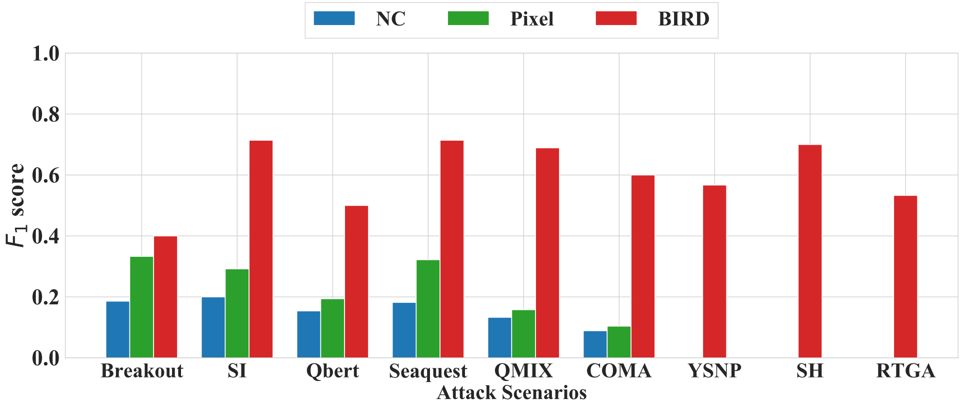

Figure S4: Backdoor detection F1 scores of the selected methods. We leverage the return to restore the trigger for BIRD and compute the F1 scores. All the other experiment settings stay the same.

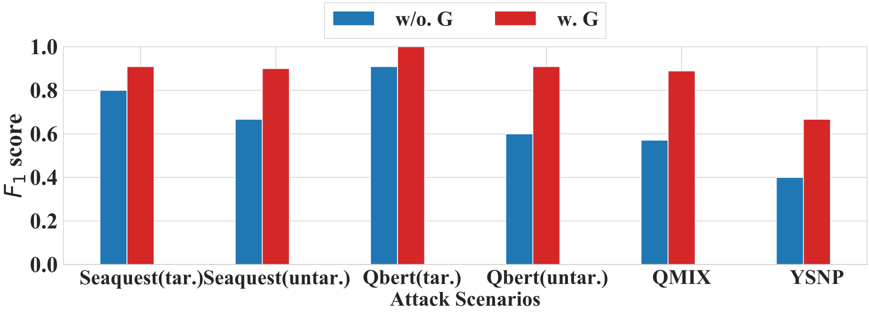

Figure S5: Ablation studies on generative model design. "tar." stands for the targeted attack and "untar." stands for the untargeted attack. We report the F1 scores for two methods.

agents, 5 agents are sub-optimal backdoored agents. For all the backdoored agents, we only use the successfully attacked ones. That is, their reward will drop to almost zero after observing the trigger. Otherwise, we treat it as an unsuccessful attack. We mix them together and then apply BIRD to detect the backdoored agents. The result is shown in Table S7. As we can see, with the increase of ϵ, we are able to capture more and more sub-optimal backdoored agents. But we will not include any false positives. Thus it is encouraged to not set ϵ as an overly regressive value in case there are sub-optimal agents that are backdoored.

We acknowledge that selecting the optimal ϵ can be a trick task especially when the sub-optimal and optimal backdoored agents are mixed together. As such, we design our final removal step with the consideration that it will not minimize the effect on the agent's performance in a clean environment. With that said, conservatively, we can apply the restoration and removal step to all the given agents. In this experiment, as shown in Table **??**, after applying the removal step to all the 20 agents, we can observe that (1) For clean agents, their performance is only marginally affected; (2) For the backdoored agent, their performance in the clean environment keep similar, and their backdoor is removed. This shows that even our detection step fails to capture some backdoored agents. With a conservative solution, the removal step can still remove the backdoor for both optimal and sub-optimal agents.

S4 Computational Efficiency and Hyper-parameter Sensitivity

Runtime comparison. On average, for one policy, the average runtime for BIRD trigger restoration is 0.75 hours, and the retraining stage takes 3.5 hours on a single NVIDIA Quadro RTX 6000 GPU. For the Provable Defense, we need to perform single value decomposition on a state matrix with about

Table S7: Performance of BIRD on detecting sub-optimal clean and backdoored agents.

ϵ	Optimal clean (5)	Suboptimal clean (5)	Optimal backdoor	Suboptimal backdoor
-0.5	5	5	5	5
-0.6	5	5	5	5
-0.7	5	5	5	4
-0.8	5	5	5	4
-0.9	5	5	5	3

Table S8: Performance of BIRD backdoor removal on 20 agents in SpaceInvaders targeted attack. We report the mean±std of the reward at clean and poisoned environments respectively. Orin. represents the original version of the policy, i.e., before applying BIRD.

Agent	Clean Env		Poisoned Env	
	Orin.	BIRD	Orin.	BIRD
Optimal clean (5)	667±74	639±104	652±81	636±98
Suboptimal clean (5)	311±96	338±83	305±97	322±74
Optimal backdoor (5)	621±73	551±86	0±0	609±82
Suboptimal backdoor (5)	342±75	312±94	22±93	308±88

30000 dimension, depends on the length of the trajectories, which takes 3.5 hours on average. Then during testing, we need to project every time step's state to another space, and the average runtime is 4.5 hour. We believe the runtime of BIRD is reasonable as it is still within the normal range of training a DRL agent in benchmark environments.

Sensitivity to restoration and detection hyper-parameters. The hyper-parameters for our backdoor restoration approach include the weights of the two regularization terms (λ_1 and λ_2) and the detection threshold (ϵ). In Table S9, we show the F1 scores of our backdoor detection technique across various hyper-parameter settings. The default values are set to $\lambda_1 = 1e-3$, $\lambda_2 = 1e-5$, and $\epsilon = -0.9$, and we systematically vary these values to assess the sensitivity of our method to different hyper-parameters. For ϵ, when its value is too small, false positive cases may arise, but the performance of BIRD stays stable when $\epsilon \leq -0.5$, showing the robustness of our detection method.

Sensitivity to number of re-initialized neurons. Table S10 presents the impact of the number of re-initialized neurons on the efficacy of backdoor removal in the targeted and untargeted attack scenarios of the SpaceInvaders Atari game. As we can observe, when the number of re-initialized neurons is set between 25 and 30 for untargeted attack, between 20 and 25 for targeted attack, the performance of our backdoor removal remains stable. These results show the insensitivity of BIRD to the key hyper-parameter for backdoor removal and demonstrate its robustness.

Sensitivity to α. In Table S10, we show the influence of the hyper-parameter α on the backdoor detection performance. Intuitively, α plays a role in controlling the variance of our generative model $\mathbf{B}_s$. A larger value of α indicates that the restored trigger values tend to concentrate around 0 and 1, while a smaller α allows for sampling values in between to explore a wider range. We observe that the performance of BIRD stays stable when $50 \leq \alpha \leq 200$ and when it is set to 400, the F1 score begins to decrease. this decline can be attributed to excessively large values of α, which hinder the generative model from exploring a broader range of possible trigger values.

Sensitivity to ϵ_1 in Eqn.(4). In Table S11, we vary the value of ϵ_1 from 1 to 0.01, and report the mean ± standard deviation of the rewards in clean and poisoned environment for two Atari game's targeted attack, after applying BIRD.

S5 Potential Social Impact.

Reinforcement learning has recently emerged as a powerful approach across various domains in computer science, including gaming, robotics, natural language processing, and computer vision. The successful application of RL in these areas highlights its potential impact and significance. In this context, our

Table S9: Influence of hyper-parameters on backdoor detection performance.

Attack scenarios	λ_1				λ_2				ϵ			
	1e-4	1e-3	5e-3	1e-2	1e-6	1e-5	5e-5	1e-4	-0.9	-0.7	-0.5	-0.3
SI(tar.)	0.833	0.833	0.800	0.667	0.800	0.833	0.875	0.500	0.833	0.833	0.800	0.500
SI(untar.)	0.909	1.000	0.800	0.727	0.750	1.000	0.909	0.889	1.000	1.000	0.750	0.571
Qbert(tar.)	0.875	1.000	0.909	0.769	0.800	1.000	0.750	0.727	1.000	1.000	0.800	0.667
Qbert(untar.)	0.875	0.909	0.909	0.667	0.909	0.909	0.875	0.750	0.909	0.909	0.727	0.500

Table S10: Influence of hyper-parameters on targeted and untargeted attacks of SpaceInvaders(SI) game. "tar." stands for targeted attack and "untar." stands for untargeted attack. We re-initialize different number of neurons in the same model and report the average reward across 1000 game rounds in the poisoned environment. For the influence of α, we set different values for α during trigger restoration and report the F1 score.

Attack scenarios	# of re-init neurons				α			
	20	25	30	40	50	100	200	400
SI(tar.)	597	609	237	135	0.800	0.833	0.800	0.667
SI(untar.)	104	510	543	147	0.909	1.000	0.889	0.633

proposed method offers a practical and valuable tool to enhance the security and integrity of RL agents deployed in these applications. By providing an effective defense against DRL backdoor attacks, our method strengthens the overall security posture of RL systems. This is particularly relevant in scenarios where large-scale RL models are shared and utilized, as it can bolster the trust and confidence in these models. By raising the bar for attackers attempting to compromise DRL agents, our method contributes to an ongoing arms race between defenders and attackers in the field of DRL backdoor attacks. This advancement in defense techniques not only protects RL-based technologies but also drives further research and development in the field. It encourages the exploration of novel defense strategies and promotes the creation of more resilient DRL systems that can withstand malicious attacks. Ultimately, this benefits the broader community that relies on DRL-based technologies, ensuring the integrity and security of RL applications across various domains.

Table S11: Influence of ϵ_1 in Eqn.(4) on backdoor removal performance.

Attack scenarios	λ_1			
	1	0.5	0.1	0.01
Breakout(clean)	284±46	298±37	302±51	293±43
Breakout(poisoned)	260±55	269±49	258±32	270±38
Seaquest(clean)	1469±198	1719±107	1656±97	1760±98
Seaquest(poisoned)	1510±121	1686±83	1652±78	1715±114

References

[1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *ICML*, 2017.

[2] Yanjiao Chen, Zhicong Zheng, and Xueluan Gong. Marnet: Backdoor attacks against cooperative multi-agent reinforcement learning. *IEEE Transactions on Dependable and Secure Computing*, 2022.

[3] Wenbo Guo, Dongliang Mu, Jun Xu, Purui Su, Gang Wang, and Xinyu Xing. Lemna: Explaining deep learning based security applications. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2018.

[4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[5] Panagiota Kiourti, Kacper Wardega, Susmit Jha, and Wenchao Li. Trojdrl: Trojan attacks on deep reinforcement learning agents. *arXiv preprint arXiv:1903.06638*, 2019.

[6] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, 2016.

[7] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

[8] Antonin Raffin, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, and Noah Dormann. Stable baselines3, 2019.

[9] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *ICML*, 2015.

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

[11] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NeurIPS*, 1999.

[12] 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.