MARNET: BACKDOOR ATTACKS AGAINST VALUE-DECOMPOSITION COOPERATIVE MULTI-AGENT REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent works have revealed that backdoor attacks against Deep Reinforcement Learning (DRL) could lead to abnormal action selection of the agent, which may result in failure or even catastrophe in crucial decision processes. However, existing attacks only consider single-agent RL systems, in which the only agent can observe the global state and have full control of the decision process. In this paper, we explore a new backdoor attack paradigm in cooperative multi-agent reinforcement learning (CMARL) scenarios, where a group of agents coordinate with each other to achieve a common goal, while each agent can only observe the local state, e.g., StarCraft II (Vinyals et al. (2017)). In the proposed MARNet attack framework, we carefully design a pipeline of trigger design, action poisoning and reward hacking modules to accommodate the cooperative multi-agent momentums. In particular, as only a subset of agents can observe the triggers in their local observations, we maneuver their actions to the worst actions suggested by an expert policy model. Since the global reward in CMARL is aggregated by individual rewards from all agents, we propose to modify the reward in a way that boosts the bad actions of poisoned agents (agents who observe the triggers) but mitigates the influence on non-poisoned agents. We conduct extensive experiments on two classical MARL algorithms VDN (Sunehag et al. (2018)) and QMIX (Rashid et al. (2018)), in two popular CMARL games Predator Prey (Boehmer et al. (2020)) and SMAC (Samvelyan et al. (2019)). The results show that MARNet outperforms baselines extended from single-agent DRL backdoor attacks TrojDRL (Kiourti et al. (2020)) and Multitasking learning (Ashcraft & Karra (2021)) by reducing the utility under attack by as much as 100%. We apply fine-tuning as a defense against MARNet, and demonstrate that fine-tuning cannot entirely eliminate the effect of the attack.

1 INTRODUCTION

Backdoor attacks are effective and stealthy attacks against deep learning models. First implemented in classification neural networks (Gu et al. (2017); Chen et al. (2017)), backdoored models can accurately classify clean inputs but misbehave for inputs with specially-designed triggers. Recent works show that deep reinforcement learning models are also vulnerable to backdoor attacks (Yang et al. (2019); Gao et al. (2020); Kiourti et al. (2020)). A backdoored DRL agent will choose a non-optimal action or even the worst action when encountering the triggers, which may lead to severe consequences in crucial RL tasks. A line of backdoor attacks against DRL has been proposed (Kiourti et al. (2020); Wang et al. (2021); Ashcraft & Karra (2021)), but almost all existing works consider the single-agent scenario, where the agent is able to observe the global state and control the entire decision process. There is a lack of backdoor attacks in the context of multi-agent reinforcement learning (MARL), especially cooperative MARL (CMARL). As far as we are concerned, there is only one work for two-agent competitive MARL backdoor attacks (Wang et al. (2021)), which is similar to single-agent RL attacks as the two agents rival with each other, and there is only one work briefly discussing the CMARL backdoor attacks (Ashcraft & Karra (2021)).

To fill up this research gap, we explore a new paradigm of backdoor attacks in the context of CMARL, where multiple agents cooperate with each other to achieve a common goal. Compared

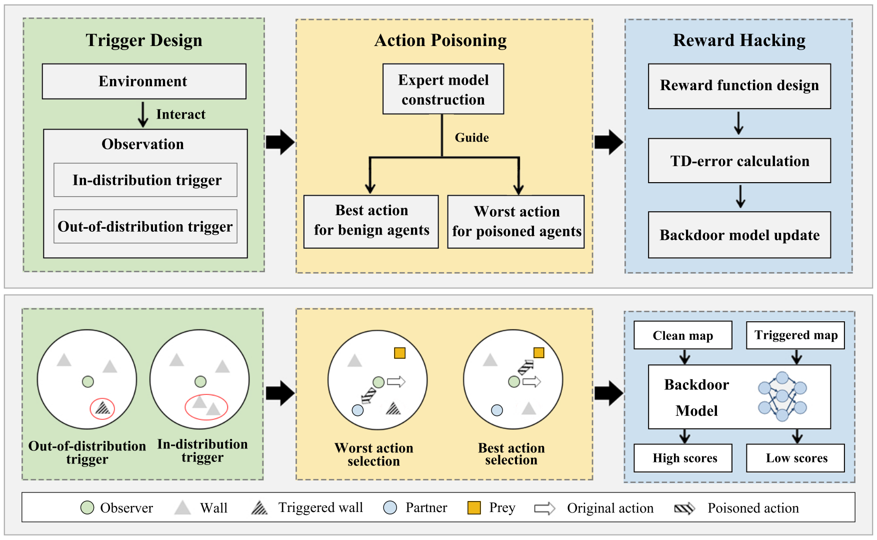


Figure 1: Overview of our proposed backdoor attack in the context of CMARL scenarios.

with backdoor attacks in single-agent RL scenarios, backdoor attacks in CMARL scenarios are faced with unique challenges.

Local observation vs. global observation. In single-agent RL scenarios, it is usually assumed that the agent is able to observe the global state. Unfortunately, in CMARL, the agents are scattered in the environments, and each agent can only observe the local state, which is a small part of the global state. The triggers used to activate the backdoor can always be perceived by the agent in the former case but are unlikely to be observed by all agents in the latter case. Popular MARL algorithms like VDN (Sunehag et al. (2018)) and QMIX (Rashid et al. (2018)) are designed to deal with local observations and are widely deployed in autonomous driving (Dosovitskiy et al. (2017); Cao et al. (2013)), robot swarms (Hüttenrauch et al. (2019)), and online video games. To the best of our knowledge, the vulnerability of these CMARL algorithms under backdoor attacks has never been investigated.

Global reward vs. individual reward. In classification tasks, backdoors are injected by changing the label of malicious inputs (inputs with triggers) to the targeted label or any wrong label. In RL tasks, backdoors are injected by hacking the rewards of actions when the environment contains triggers. By assigning a high reward to actions that are actually bad in the malicious environment, the agents will learn to choose these actions and end up in failure. In single-agent RL scenarios, the global reward is directly fed back to the agent to train the policy network. However, CMARL algorithms usually adopt the centralised training decentralised executing (CTDE) (Oliehoek et al. (2008); Kraemer & Banerjee (2016); Foerster et al. (2018)) framework, where the global reward is aggregated based on the rewards from all agents, including poisoned and non-poisoned agents, and the policy network is trained centrally and shared by all agents for action selection. The reward aggregation in CMARL complicates the backdoor injection process.

To address these challenges, we propose a novel backdoor attack strategy against value-decomposition CMARL, named MARNet, which enables agents to behave normally when the environment is clean but induces abnormal behaviors with the presence of triggers. MARNet features special designs in all of the three modules in its pipeline, namely, trigger design, action poisoning, and reward hacking.

Trigger design. Existing DRL backdoor attacks introduce the influence of triggers by directly altering the observation of the single agent (Kiourti et al. (2020)) or controlling the actions of the agent (Wang et al. (2021)). We adopt a more general and practical strategy by embedding the triggers in the environment with low visibility. We leverage both in-distribution triggers (Wang et al. (2020b); Wang et al. (2021)) and out-of-distribution triggers (Kiourti et al. (2020)) in the environment to activate the backdoor.

Action poisoning. Unlike existing methods (Kiourti et al. (2020)) that make the agent choose a specific action or a random action that is not optimal, we aim to force the agent to play the worst possible action when observing triggers to degrade the utility to a large extent. We design an action poisoning method which determines the worst possible action according to an expert policy model. The proposed method can pinpoint the worst action with much less overhead than existing works (Wang et al. (2021); Ashcraft & Karra (2021)).

Reward hacking. To accommodate the CTDE framework in CMARL scenarios, we design a new reward hacking algorithm that manipulates the global reward during centralized training. Since the global reward judges the behaviors of all agents, we differentiate the contributions of non-poisoned agents (observe no triggers) and poisoned agents (observe some triggers).

We implement and evaluate the proposed MARNet attack with extensive evaluations. We conduct experiments in two popular CMARL games, namely Predator Prey (Boehmer et al. (2020)) and SMAC (Samvelyan et al. (2019)), against two commonly-used CMARL algorithms, namely VDN (Sunehag et al. (2018)) and QMIX (Rashid et al. (2018)). Results verify that our proposed attack outperforms extensions of existing single-agent DRL backdoor attacks by reducing the utility under attack by as much as 100%. Ablation study confirms the necessity of each design module. We also demonstrate that common defense strategy fine-tuning can mitigate the utility drop but cannot eliminate the attack effect.

Our work reveals the vulnerability of CMARL algorithms to backdoor attacks, which may spur research in related areas to improve the security of CMARL algorithms in critical applications, e.g., autonomous driving.

2 BACKGROUND

2.1 MULTI-AGENT REINFORCEMENT LEARNING

Reinforcement learning (RL) solves problems that can be formulated as Markov Decision Processes (MDP) $\langle S, A, P, r, \gamma \rangle$, where S is the state space, A is the action space, P is the state transition probabilities, r is the reward function, and $\gamma \in [0, 1)$ is a discount factor. With unknown state transition probabilities, the agent interacts with the environment to gradually learn an optimal policy π that can maximize the total accumulative reward (i.e., utility).

Deep Reinforcement Learning (DRL) is proposed to deal with exorbitantly large state spaces of complicated MDPs. Deep Neural Networks (DNN) are used to represent the agent's functions (e.g. value function, Q-value function, and policy function). Deep Q-Learning (Mnih et al. (2015)) extends the traditional Q-learning algorithm by using a neural network to represent the Q-value $Q(s, a)$ that estimates the accumulative reward at state s when taking action a. The Q-network is trained and updated by minimizing the TD error using the ϵ-greedy approach.

$$\mathbf{L}(\theta) = \sum_{i=1}^{b} (y_i - Q(s, a; \theta))^2), \tag{1}$$

where θ represents the parameters of the Q-network, b is the number of sample batches of the replay memory, and $y_i = r + \gamma \max_{a'} Q(s', a'; \theta^-)$, in which θ^- represents the parameters of a target network periodically copied from θ.

Multi-agent reinforcement learning (MARL) emerges to tackle partially observable Markov Decision Process (POMDP) problems, where a single agent cannot observe the panorama of the entire environment. In this case, multiple agents cooperate to make decisions based on cooperative multi-agent reinforcement learning (CMARL), which can be formulated as a Dec-POMDP tuple $G = \langle S, U, P, r, Z, O, n, \gamma \rangle$ (Oliehoek & Amato (2016)), where S represents the global state of the environment, U is the set of actions of all agents, Z is the set of individual observations of all agents determined by the observation transition function $O(s, a)$, and n is the number of all agents in the environment. P, r and γ represent state transition probabilities, reward, and discounted factor as in traditional MDP. Since it is difficult to derive the optimal policy under incomplete observation, the action-observation history τ of agents is usually gathered in POMDP to get more state information in many CMARL algorithms (Peng et al. (2017); Tampuu et al. (2017); Hausknecht & Stone (2015)).

To deal with partial observation and multiple agents, existing CMARL algorithms adopt the centralised training and decentralised executing (CTDE) framework as shown in Appendix A.1. In the training phase, a single model is trained with information gathered from all agents. The trained model is then distributed and shared by all agents. In the execution phase, each agent selects its action according to the observation and the shared model. The CTDE framework can be improved with communications between agents, parameter sharing, and value-decomposition (Foerster et al. (2016); Terry et al. (2020)).

Most of the existing CMARL algorithms can be classified into value-based, actor-critic and value-decomposition methods. A widely-used value-based CMARL algorithm is Independent Q-Learning (IQL) Tan (1993), in which each agent regards other agents as a component of the environment and acts as non-cooperative individuals to achieve cooperation. However, IQL cannot guarantee convergence due to the unstable environment induced by different policies of individual agents. Another line of MARL algorithms is based on the actor-critic framework, e.g., COMA (Foerster et al. (2018)) and VDAC (Su et al. (2020)), using a centralized critic to judge the actions of all agents. Nonetheless, training a centralized critic may be infeasible with a huge number of agents. In comparison, value-decomposition CMARL has ideal convergence and scalability properties. The rationale is to decompose the global Q-value $Q_{total}(s, u)$ into the Q-values of individual agents. The Q-value of each agent $Q_i(\tau_i, a_i)$ is computed by itself based on its partial observation history τ_i and action a_i. Then, the individual Q-values are aggregated into $Q'_{total}(\tau, u)$ to approximate the real Q_{total}, where τ is the set of the action-observation history of all agents, and u is the current action set. Q'_{total} is used to update the centralized model. In value-decomposition CMARL, the key is to estimate Q'_{total} in a way that can closely approximate the true Q_{total}. An early algorithm VDN sums up the Q-values of all agents (Sunehag et al. (2018)).

$$\mathbf{Q'_{total}} = \sum_{i=1}^{n} Q_i. \tag{2}$$

QMIX (Rashid et al. (2018)) replaces the linear operation in VDN with a mixing neural network to approximate Q_{total}. QMIX imposes the constraint $\frac{\partial Q'_{total}}{\partial Q_i} > 0, \forall i \in [1, n]$ to make sure that the local best action conforms to the global best action set. A hypernetwork is adopted to generate the parameters of the mixing network with the input of the global state of the environment. Therefore, the Q'_{total} function in QMIX contains s and is represented as $Q'_{total}(\tau, u, s)$. Other algorithms, e.g., QTRAN and WQMIX (Rashid et al. (2020)), are extensions of VDN or QMIX.

2.2 Backdoor Attacks against Deep Reinforcement Learning

Backdoor attacks are first proposed for deep neural networks (Gu et al. (2017)), where the backdoored model can accurately identify clean samples but misclassify malicious samples with a specially-designed trigger. Apart from the computer vision domain (Liu et al. (2018); Salem et al. (2020); Gao et al. (2021)), backdoor attacks are also effective in natural language processing (Dai et al. (2019); Chen et al. (2020)), signal classification (Davaslioglu & Sagduyu (2019)), and deep reinforcement learning.

In the context of deep reinforcement learning, the aim of backdoor attacks is to degrade the utility of the agent as much as possible via injecting backdoors into their policy networks. Let $\mathcal{U}(\pi, \mathcal{V})$ denote the expected utility of the agent by adopting policy π in an environment $\mathcal{V}$. The backdoor attacks against DRL intend to substitute a normal policy π with a backdoored policy $\widehat{\pi}$ such that the expected utility of adopting $\widehat{\pi}$ in a clean environment $\mathcal{V}$ is close to that of adopting π in $\mathcal{V}$,

$$|\mathcal{U}(\pi, \mathcal{V}) - \mathcal{U}(\widehat{\pi}, \mathcal{V})| \leq \delta, \tag{3}$$

and the expected utility of adopting $\widehat{pi}$ in a malicious environment $\mathcal{V} + \mathcal{T}$ with trigger $\mathcal{T}$ is as worse as possible,

$$\max_{\widehat{\pi}} \ \mathcal{U}(\pi, \mathcal{V} + \mathcal{T}) - \mathcal{U}(\widehat{\pi}, \mathcal{V} + \mathcal{T}). \tag{4}$$

The attack surface of backdoor attacks in DRL can be extrinsic or intrinsic. The environment and the reward are extrinsic to the agent and can be easily altered by the attacker. The policy network structure and the training algorithm are intrinsic to the agent and are more difficult to access by the attacker. In white-box attacks, both extrinsic and intrinsic attack surfaces are available, while in black-box attacks, only extrinsic attack surfaces are available.

Existing backdoor attacks against DRL mainy adopt two strategies, i.e., *non-optimal action strategy* and *worst action strategy*. Non-optimal action strategy aims to divert the agent from the optimal action to a specific action (targeted attack) or a random action (untargeted attack) to indirectly degrade the policy and decrease the utility. TrojDRL (Kiourti et al. (2020)) is a non-optimal action attack considering the black-box scenarios. Wang et al. (Wang et al. (2020b)) proposed a non-optimal action attack, Stop-and-Go, for black-box traffic congestion control systems by designing the trigger as a combination of sensor measurements and inserting malicious state-action pairs in the training dataset. Different from the non-optimal action strategy, the worst action strategy aims to drive the agent to the worst possible action under the current state to minimize the utility. Ashcraft et al. (Ashcraft & Karra (2021)) designed a worst action attack using multitask learning to inject the backdoor. However, the training process needs to create an entirely different poisoned environment, which requires a particular environment configuration. BackdooRL (Wang et al. (2021)) is a worst action attack that considers competitive two-agent MARL rather than cooperative MARL. BackdooRL uses imitation learning to compute the worst action when the trigger (a series of predefined actions played by the attacker) is present. Due to the high complexity of imitation learning, the attack requires more than triple the training overheads of conventional attacks against DRL.

Our proposed backdoor attack is against cooperative MARL (CMARL), and we focus on value-decomposition algorithms, which are widely adopted in CMARL problems. We design a *worst action strategy* that affects all cooperative agents and slash their overall utility.

2.3 Threat Model

In this paper, we focus on CMARL, which is common in the game domain (e.g., StarCraft II (Vinyals et al. (2017))), where multiple agents cooperatively act to win the game for the player. We assume that the attacker is a malicious game adapter and the victims are game players. The attacker utilizes the plugin provided by the game developer (e.g., Steam Workshop[1]) to adapt the game scenarios with triggers and upload the adapted game scenarios to the game community. The game players usually acquire a well-trained model from the game developer or the game community to help them play games. For the customized game scenarios adapted by the malicious attacker, the model provided by the attacker may better fit the adapted scenarios than the original model provided by the game developer. The attacker lures players to download their backdoored model, enabling the agents to perform the right actions to win under clean game maps (environment), but deflects the agents to a crushing loss under manipulated game maps with the triggers.

3 Methodology

As shown in Figure 1, our proposed backdoor attack, named MARNet, proceeds through three phases: trigger design, action poisoning, and reward hacking. To begin with, we utilize both out-of-distribution and in-distribution triggers to alter the environment. Then, we develop an expert policy model to poison the actions of agents who observe the triggers in random steps. Finally, we hack the global reward and obtain the backdoored policy model.

3.1 Trigger Design

The trigger in traditional RL backdoor attacks can be classified into two categories, in-distribution triggers (Wang et al. (2020b); Wang et al. (2021)) and out-of-distribution triggers (Kiourti et al. (2020)). In-distribution triggers are formed by internal components of the environment, while out-of-distribution triggers are formed by external components. In-distribution triggers are more stealthy than out-of-distribution triggers (Ashcraft & Karra (2021)), but are less learnable than out-of-distribution triggers since in-distribution triggers are similar to other elements in the environment. In MARNet, we use both in-distribution and out-of-distribution triggers to show the generality of our attack.

[1]http://steamcommunity.com/workshop/

Instead of directly tampering with the observation of agents in most existing works (Kiourti et al. (2020); Wang et al. (2020a)), we make subtle changes to the environment (e.g., maps) with triggers as shown in Figure 5 in Appendix A.2. For in-distribution triggers, we slightly modify existing components in the environment, e.g., terrain height or ground texture. For out-of-distribution triggers, we apply a small imperceptible pattern Δ to the environment (Kiourti et al. (2020)). To ensure stability, the triggers are designed as stationary objects (e.g., walls, trees, or terrain) rather than movable objects (e.g., preys).

3.2 Action Poisoning

The triggers added to the environment can only be observed by a subset of agents since no agent has a global view of the entire environment. We only poison the actions of the agents if at least one agent has observed the triggers. This guarantees that the backdoored policy network will behave as a normal policy network when there is no trigger as required by backdoor attacks.

For poisoned agents, we adopt the worst action strategy, which reduces the utility of the player more than the non-optimal action strategy. The worst action strategy of BackdooRL (Ashcraft & Karra (2021)) is based on imitation learning (Ho et al. (2016)), which involves a complicated multi-shot training process. The behavior cloning of imitation learning in BackdooRL makes the backdoored policy model learn the trajectories of the expert model. Nevertheless, a manual-operation hardcoded model is needed to generate trajectories and to minimize the distance between the hardcoded model and the training model. The worse action strategy in backdoor attacks proposed by (Ashcraft & Karra (2021)) utilized multi-task learning, but it needs a completely different poisoned environment for training.

We design an efficient and lightweight action poisoning method for MARNet. We first train a normal policy model in a clean environment, which we call the expert model. Then, during the training of the backdoored policy model, in each poisoned step (randomly chosen), we utilize the expert model to infer the worst action that has the the minimum probability for the poisoned agents. For other agents, we choose the best action suggested by the expert model. In non-poisoned steps, we follow the normal training process. The action poisoning is summarized as follows.

$$a_{poisoned} = \begin{cases} \mathbf{argmax}_a(D(\tau, u; \pi_{expert})), & \text{non-poisoned agent,} \\ \mathbf{argmin}_a(D(\tau, u; \pi_{expert})), & \text{poisoned agent,} \end{cases} \tag{5}$$

$$a_{clean} = \epsilon - greedy(\mathbf{argmax}_a(D(\tau, u; \pi_{train})), \epsilon), \tag{6}$$

where $a_{poisoned}$ and a_{clean} are the actions chosen in poisoned step and non-poisoned steps respectively during training, D is the action distribution of the expert policy π_{expert} given the observation history τ and the action history u, ϵ is the probability of ϵ-greedy action selection.

Choosing the best actions for non-poisoned agents in poisoned steps according to the expert model can maintain an ideal performance in a clean environment. It also aligns with the reward calculation process of value-decomposition CMARL algorithms, which we will introduce in the next section.

3.3 Reward Hacking

To ensure that the selected poisoned actions will be learned by the agent, we need to increase their reward to make the agent favor the selected poisoned actions in the malicious environment with the triggers. Different from existing backdoor attacks that only consider the reward of one agent in single-agent RL, MARNet deals with the CMARL scenario where the rewards of all agents are aggregated for centralised training. To address this issue, we design a new reward hacking method called mixing global reward hacking (MGR).

In value-decomposition CMARL, the global reward is used to update the policy model parameter according to equation (1). In CMARL systems, not every agent can observe the triggers. Instead of setting the reward of selecting the modified actions as the maximum as in trojDRL, we should only hack the global rewards in a way that increases the rewards of agents whose actions are poisoned (agents who have observed the triggers) and tries to maintain the rewards of other agents.

trigger size	VDN						QMIX					
	0%	5%	10%	15%	20%	25%	0%	5%	10%	15%	20%	25%
Normal model	31.21	32.00	31.57	30.07	31.18	30.57	32.90	32.46	33.10	32.03	32.48	32.91
MTL	31.75	32.06	31.56	31.88	30.77	30.70	31.92	31.91	31.53	31.62	30.56	30.25
TrojDRL	27.03	25.55	16.52	-1.88	-20.64	-42.34	28.41	23.71	10.28	-5.50	-33.44	-67.51
Ours	**30.60**	**19.14**	**11.63**	**-9.25**	**-24.25**	**-44.05**	**32.20**	**23.30**	**12.02**	**1.32**	**-22.75**	**-70.00**

Table 1: Predator Prey scores with different attack models against different CMARL algorithms. A lower score indicates that the attack is more effective. A 0% trigger size represents a clean map.

$$\mathbf{R}_{hacked} = r_{max} \times n_p + \frac{n - n_p}{n} \times R^{-} + \frac{1}{L_{max}} \times R_{win}, \tag{7}$$

where r_{max} is the maximum per-step reward, n_p is the number of poisoned agents, n is the number of agents, R^{-} is the original global reward, R_{win} is the reward for winning the game, L_{max} is the expected maximum length of the game. In (7), we aim to raise the reward of each poisoned agent to the maximum but preserve the local reward of non-poisoned agents. In addition, if the state is winning and the reward is a final winning reward, we divide the winning reward by the maximum length of the game to approximate the final accumulated reward.

We summarize the complete algorithm of MARNet in Algorithm 1 in the Appendix.

4 EXPERIMENT

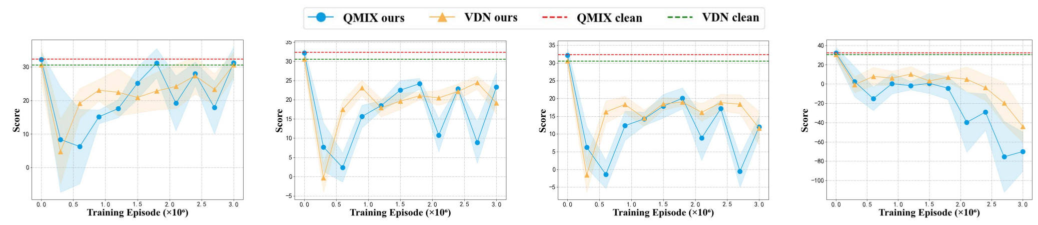
To show the effectiveness of MARNet, we conduct attacks against two classical CMARL algorithms, namely VDN (Sunehag et al. (2018)) and QMIX (Rashid et al. (2018)). We experiment in two popular CMARL games, namely Predator Prey (Boehmer et al. (2020)) and SMAC (Samvelyan et al. (2019)).

Predator Prey. Predator Prey is a CMARL game where the agents coordinate to capture as many preys as possible. The preys, including stags and hares, must be captured by more than two agents. A reward of 10/1 is gained for each capture of stag/hare. If agents collide, a reward of -0.1 is obtained as a punishment. The map is a 10×10 grid, and the observation range of each agent is 5×5. We randomly place eight agents, eight stags, eight hares, and 25 walls in the initial state.

SMAC. SMAC provides a CMARL environment that contains classic micro StarCraft II scenarios. The allied units controlled by the player compete with the enemy units controlled by the computer. A reward of 10 is gained if the allied units kill an enemy. If all enemy units are killed (win), a reward of 200 is gained. If all allied units are killed (lose), a reward of 0 is gained. We have tested four different maps in SMAC.

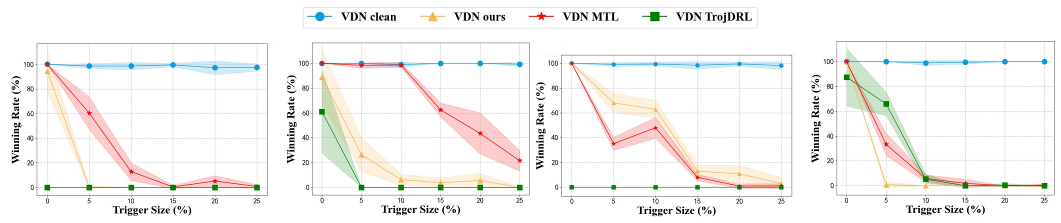
We implement two baselines. The first baseline adapts the multi-task learning (MTL) in (Ashcraft & Karra (2021)) for the cooperative MARL scenarios. The second baseline extends TrojDRL (Kiourti et al. (2020)) to CMARL. We conduct an ablation study to evaluate the effectiveness of action poisoning and reward hacking modules in MARNet. The main evaluation metric is the utility in both the clean and the triggered environments. In Predator Prey, the utility of the player is the final score, and in SMAC, the utility of the player is the winning rate against the computer that adopts a *super hard* model to play.

Our experiments run on a machine with an Intel(R) Xeon(R) Gold 5117 CPU and Nvidia GeForce RTX 3080 GPUs. The operating system is Ubuntu 18.04.1 LTS. In Predator Prey, we train the clean policy model for 1,050,000 episodes, and the backdoored policy models are trained for another 2,100,000 episodes based on the clean policy model. In SMAC, we train the clean policy model for 10,050,000 episodes, and the backdoored policy models are trained for another 10,050,000 episodes based on the clean policy model. The expert model used in action poisoning is just the clean policy model.



(a) Clean (b) 5% triggers (c) 10% triggers (d) 25% triggers

Figure 2: Predator Prey scores at different training episodes.



(a) VDN in 2s3z (b) QMIX in 2s3z (c) VDN in 8m (d) QMIX in 8m

Figure 3: SMAC winning rate in $2s3z$ and $8m$, two StarCraft II maps.

4.1 Effectiveness of Attacks

Predator Prey. The triggers in Predator Prey (PP) are designed as out-of-distribution triggers, i.e., adding a pattern Δ into some of the walls in the map as shown in Figure 5. As shown in Table 1, MARNet achieves nearly the same score as the normal model when the environment contains no triggers (0%). When there are triggers, compared to MTL, our attack reduces the score of the player by more than 30% when the trigger size is as small as 5%, and decreases the score by as much as 300% when the trigger size is 25%. As the score keeps decreasing sharply with larger trigger sizes by our attack, the scores of MTL hardly change. The poisoned environment we design for MTL is to inverse the reward of the original environment. TrojDRL decreases the scores in triggered maps more than MTL, but TrojDRL degrades the utility in the clean map, which is unacceptable for backdoor attacks. This indicates that directly extending backdoor attacks of single-agent DRL to CMARL is ineffective. Figure 2, shows the scores with different trigger sizes at different training episodes. At the beginning, the score sharply drops. With more training episodes, the score gradually fluctuates but finally converges to a low value. In the clean map, the score maintains the same as the normal policy model at episode 0.

SMAC. The triggers in SMAC are designed as in-distribution trigger as shown in Figure 5. As shown in Figure 3 and Figure 6 in the Appendix, MARNet outperforms the two baselines in most of the maps. TrojDRL performs poorly in QMIX and even lose effectiveness in VDN. The poisoned environment we design for MTL is to swap the reward of win and defeat and the reward of killing an enemy and losing an ally. MTL behaves unstably in different maps. The possible reason is that the effectiveness of MTL depends on the setting of the poisoned tasks, and it is hard to figure out the best environment parameters. As for MARNet, with a trigger size of 5%, the winning rate of VDN drops from nearly 100% to 0% in the $2s3z$. The winning rate of QMIX drops from 90% to 25%. As for other maps, MARNet still keeps higher performance and robustness than the two baselines.

4.2 Ablation Study

Action poisoning. To verify the effectiveness of the proposed action poisoning strategy, we conduct an ablation study in Predator Prey using the QMIX algorithm. We compare our action poisoning strategy with the targeted action modification and untargeted action modification of TrojDRL. In the targeted attack, we choose action *stay* as the targeted action. As shown in Table 2, adopting the action poisoning strategy of the targeted and untargeted attacks of trojDRL reduces the score obtained in the malicious environment more than our attack. However, the score in the clean environment (a

	Action poisoning	Reward hacking	Trigger size					
			0%	5%	10%	15%	20%	25%
Clean model	-	-	32.90	32.46	33.10	32.03	32.48	32.91
TrojDRL[1]	Targeted	MGR	18.00	-5.55	-35.74	-80.48	-111.77	-122.90
TrojDRL[2]	Untargeted	MGR	24.45	23.34	4.10	-14.55	-46.37	-79.00
Max_reward	EGA	Max value	10.60	8.35	5.85	2.72	-1.11	-12.00
Same_reward	EGA	No change	33.00	25.91	23.50	19.37	14.52	11.78
Ours	EGA	MGR	**32.20**	**23.30**	**12.02**	**1.32**	**-20.75**	**-70.00**

Table 2: Ablation study of action poisoning and reward hacking modules. TrojDRL[1] and TrojDRL[2] are targeted and untargeted versions of trojDRL.

	Trigger size					
	0%	5%	10%	15%	20%	25%
Clean model	32.90	32.46	33.10	32.03	32.48	32.91
Fine-tune[1]	32.15	30.25	25.70	23.50	19.85	10.50
Fine-tune[2]	32.63	30.05	28.59	24.95	22.15	21.30
No Fine-tune	32.20	23.30	12.02	1.32	-20.75	-70.00

Table 3: Predator Prey scores using the QMIX algorithm. We fine-tune our backdoored model for 1,000,000 episodes (Fine-tune[1]) and 2,000,000 episodes (Fine-tune[2]) respectively.

trigger size of 0%) using TrojDRL action poisoning strategies is much lower than the clean model. This means that the TrojDRL action poisoning strategies degrade the performance in the clean environment, which is unacceptable for backdoor attacks. In comparison, our attack can maintain a high score in clean environments.

Reward hacking. We compare our proposed reward hacking algorithm, i.e., mixing global reward (MGR), with the reward hacking algorithms of TrojDRL and Stop-and-Go. In TrojDRL, the reward of the poisoned agent is assigned the maximum value. In Stop-and-Go, there is no change to the reward of the poisoned agent. As shown in Table 2, the reward hacking methods in existing works cannot effectively degrade the utility of the player since they did not consider the reward aggregation process in CMARL. Our proposed MGR method adapts the global reward in regard to both poisoned and non-poisoned agents, attaining an ideal effect in CMARL.

4.3 Resistance to Defense

Existing defense methods for backdoor attacks are mostly designed for deep neural networks (Gu et al. (2019); Liu et al. (2018); Ji et al. (2018); Salem et al. (2020)), and there is a lack of defense for backdoor attacks against DRL. A potential way of defending against backdoor attacks against DRL is by fine-tuning, where the defender retrains the backdoored policy model in clean environments. We evaluate the fine-tuning strategy in the Predator Prey environment using the QMIX algorithm. Table 3 illustrates that fine-tuning can mitigate the attack power to some extent but cannot entirely compensate for the utility drop. The scores of fine-tuned backdoored policy models are still lower than the normal model in the presence of triggers.

5 Conclusion

Our work has concluded that it is feasible to insert a backdoor into multi-agent reinforcement learning models. We present a new backdoor attack method against value-decomposition cooperative MARL and implement our method in two classical value-decomposition CMARL algorithms, i.e., VDN and QMIX. The results show that our attack can make backdoored CMARL models behave well in normal scenarios but quickly deteriorate in malicious scenarios with triggers formulated by the attacker. We try the commonly-used defense for backdoor attacks and discover that fine-tuning cannot completely remove the effect of the attack, which notes the vulnerability of existing CMARL algorithms to backdoor attacks.

6 Ethics Statement

Broader Impact. Our research will raise concerns about the security of cooperative multi-agent reinforcement learning (CMARL) and spur research on defending against backdoor attacks in CMARL. Since the traditional backdoor defenses such as fine-tuning are ineffective, designing more advanced backdoor defenses for CMARL will attract more social attention.

Negative Impact. Our research may be leveraged by attackers to actually launch such backdoor attacks against cooperative multi-agent reinforcement learning, which will cause potential economic loss. Our research may be exploited to develop more powerful backdoor attacks against CMARL.

REFERENCES

Chace Ashcraft and Kiran Karra. Poisoning deep reinforcement learning agents with in-distribution triggers. *arXiv preprint arXiv:2106.07798*, 2021.

Wendelin Boehmer, Vitaly Kurin, and Shimon Whiteson. Deep coordination graphs. In *International Conference on Machine Learning*, pp. 980–991, 2020.

Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial Informatics*, 9(1): 427–438, 2013.

Xiaoyi Chen, Ahmed Salem, Michael Backes, Shiqing Ma, and Yang Zhang. Badnl: Backdoor attacks against nlp models. *arXiv preprint arXiv:2006.01043*, 2020.

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.

Jiazhu Dai, Chuanshuai Chen, and Yufeng Li. A backdoor attack against lstm-based text classification systems. *IEEE Access*, 7:138872–138878, 2019.

Kemal Davaslioglu and Yalin E Sagduyu. Trojan attacks on wireless signal classification with adversarial machine learning. In *IEEE International Symposium on Dynamic Spectrum Access Networks*, pp. 1–6, 2019.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Annual Conference on Robot Learning*, volume 78, pp. 1–16, 2017.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Jakob N Foerster, Yannis M Assael, Nando De Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. *arXiv preprint arXiv:1605.06676*, 2016.

Yansong Gao, Bao Gia Doan, Zhi Zhang, Siqi Ma, Jiliang Zhang, Anmin Fu, Surya Nepal, and Hyoungshick Kim. Backdoor attacks and countermeasures on deep learning: A comprehensive review. *arXiv preprint arXiv:2007.10760*, 2020.

Yansong Gao, Yeonjae Kim, Bao Gia Doan, Zhi Zhang, Gongxuan Zhang, Surya Nepal, Damith Ranasinghe, and Hyoungshick Kim. Design and evaluation of a multi-domain trojan detection method on deep neural networks. *IEEE Transactions on Dependable and Secure Computing*, 2021.

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

Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. BadNets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.

Matthew J. Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents*, pp. 29–37, 2015.

Jonathan Ho, Jayesh Gupta, and Stefano Ermon. Model-free imitation learning with policy optimization. In *International Conference on Machine Learning*, volume 48, pp. 2760–2769, 2016.

Maximilian Hüttenrauch, Sosic Adrian, Gerhard Neumann, et al. Deep reinforcement learning for swarm systems. *Machine Learning Research*, 20(54):1–31, 2019.

Yujie Ji, Xinyang Zhang, Shouling Ji, Xiapu Luo, and Ting Wang. Model-reuse attacks on deep learning systems. In *ACM SIGSAC Conference on Computer and Communications Security*, pp. 349–363, 2018.

Panagiota Kiourti, Kacper Wardega, Susmit Jha, and Wenchao Li. Trojdrl: Evaluation of backdoor attacks on deep reinforcement learning. In *ACM/IEEE Design Automation Conference*, pp. 1–6, 2020.

Landon Kraemer and Bikramjit Banerjee. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, 190:82–94, 2016.

Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. In *Annual Network and Distributed System Security Symposium*. The Internet Society, 2018.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Frans A Oliehoek and Christopher Amato. *A concise introduction to decentralized POMDPs*. Springer, 2016.

Frans A Oliehoek, Matthijs TJ Spaan, and Nikos Vlassis. Optimal and approximate q-value functions for decentralized pomdps. *Artificial Intelligence Research*, 32:289–353, 2008.

Peng Peng, Ying Wen, Yaodong Yang, Quan Yuan, Zhenkun Tang, Haitao Long, and Jun Wang. Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games. *arXiv preprint arXiv:1703.10069*, 2017.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, volume 80, pp. 4295–4304, 2018.

Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. Weighted qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 10199–10210, 2020.

Ahmed Salem, Rui Wen, Michael Backes, Shiqing Ma, and Yang Zhang. Dynamic backdoor attacks against machine learning models. *arXiv preprint arXiv:2003.03675*, 2020.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philiph H. S. Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft multi-agent challenge. *CoRR*, abs/1902.04043, 2019.

Jianyu Su, Stephen C. Adams, and Peter A. Beling. Value-decomposition multi-agent actor-critics. In *AAAI Conference on Artificial Intelligence*, pp. 11352–11360, 2020.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *International Conference on Autonomous Agents and Multi-Agent Systems*, pp. 2085–2087, 2018.

Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PloS One*, 12(4):e0172395, 2017.

Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *International Conference on Machine Learning*, pp. 330–337. 1993.

Justin K Terry, Nathaniel Grammel, Ananth Hari, and Luis Santos. Parameter sharing is surprisingly useful for multi-agent deep reinforcement learning. *arXiv e-prints*, pp. arXiv–2005, 2020.

Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017.

Lun Wang, Zaynah Javed, Xian Wu, Wenbo Guo, Xinyu Xing, and Dawn Song. Backdoorl: Backdoor attack against competitive reinforcement learning. In *International Joint Conference on Artificial Intelligence*, 2021.

Shuo Wang, Surya Nepal, Carsten Rudolph, Marthie Grobler, Shangyu Chen, and Tianle Chen. Backdoor attacks against transfer learning with pre-trained deep learning models. *IEEE Transactions on Services Computing*, 2020a.

Yue Wang, Esha Sarkar, Michail Maniatakos, and Saif Eddin Jabari. Watch your back: Backdoor attacks in deep reinforcement learning-based autonomous vehicle control systems. *Work*, 8(28): 12, 2020b.

Zhaoyuan Yang, Naresh Iyer, Johan Reimann, and Nurali Virani. Design of intentional backdoors in sequential models. *arXiv preprint arXiv:1902.09972*, 2019.

A Appendix

A.1 Centralised Training and Decentralised Executing

Centralised Training and Decentralised Executing (CTDE) framework is widely adopted in multi-agent problems to learn a policy. Centralised training gathers the data of all agents for training, overcoming the difficulty that the agents cannot observe the whole state. The trained model is shared by every agent when executing. Different MARL algorithms adopt different CTDE frameworks, but most of them follow the process as shown in Figure 4.

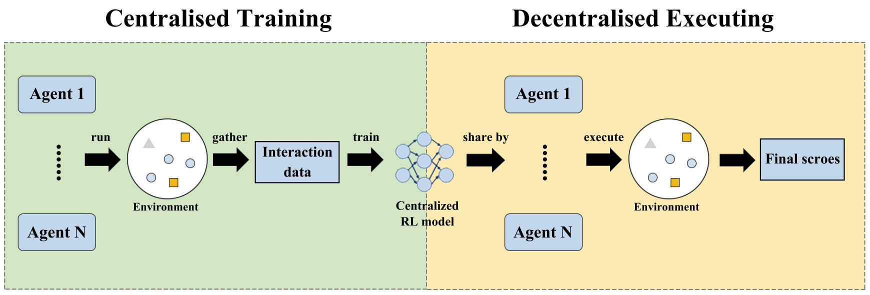


Figure 4: The main process of centralised Training and Decentralised Executing.

A.2 Trigger Design

Figure 5 shows the out-of-distribution and in-distribution trigger patterns in the environments. In the map of Predator Prey, which is described as a grid, the modifications we apply do not exist in the natural environments, i.e., out-of-distribution triggers. In the map of SMAC, we modify the terrain height or the local texture, which are components of the map of SMAC, i.e., in-distribution triggers.

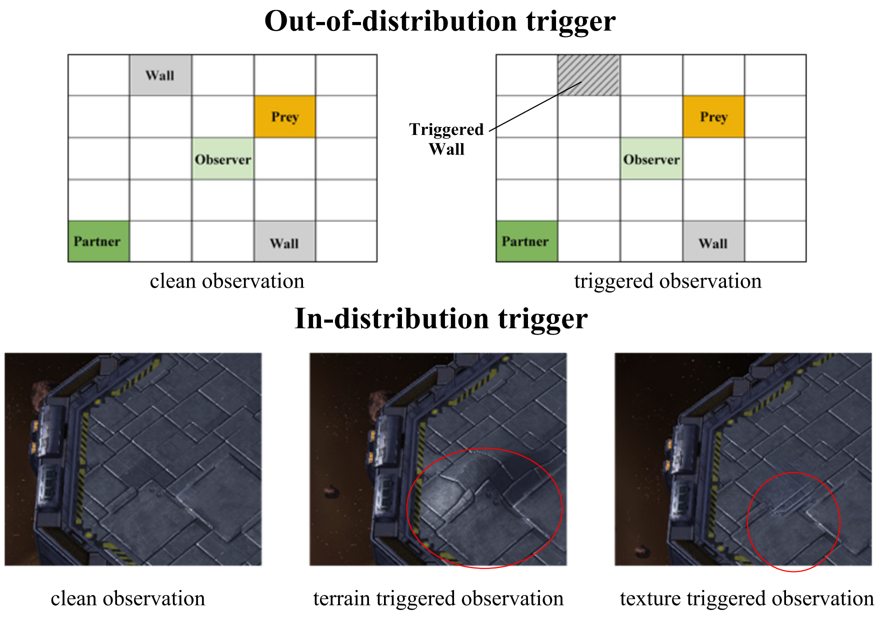
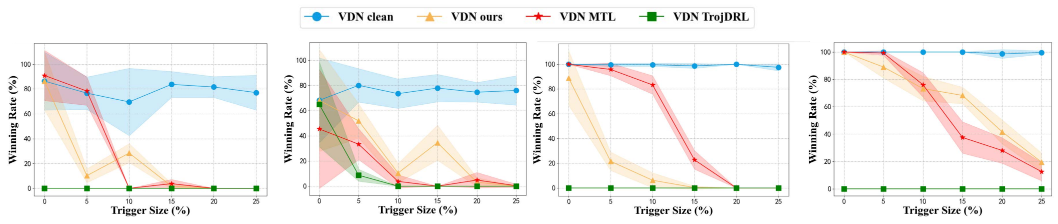


Figure 5: Out-of-distribution and in-distribution triggers.



(a) VDN in 5m_vs_6m (b) QMIX in 5m_vs_6m (c) VDN in 3s4z (d) QMIX in 3s4z

Figure 6: SMAC winning rate in *5m_vs_6m* and $3s4z$, two StarCraft II maps.

Algorithm 1 Backdoor Attack against value-decomposition cooperative MARL

Require: Network of expert model M, network of the pre-trained model m, the environment of the game E, the maximum number of iteration in training T, the number of the agents N, the learning batch size b, the poison rate p, and the learning rate lr.

Ensure: Network of backdoor model m

```
1:  initial data buffer $B$ for storing the training data
2:  for i from 0 to $T$ do
3:      $data$ = run$(m, E)$.
4:      insert $data$ into $B$
5:      if the size of $B > b$ then
6:          $sample$ = $B$.sample$(b)$.
7:          if with the probability $p$ then
8:              for i, j in batch, episode from $sample$ do
9:                  Randomly choose agents $A_p$ to poison. The rest of the agents is $A_c$.
10:                 poison_obs$(A_p, sample[i, j])$
11:                 choose_best_actions$(M, A_p, sample[i, j])$
12:                 choose_worst_actions$(M, A_p, sample[i, j])$
13:                 Hack reward in $sample$ as equation (7)
14:             end for
15:         end if
16:         update $m$ with the $sample$
17:     end if
18: end for
19: return $m$.
```