This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3207429

MARNet: Backdoor Attacks against Cooperative Multi-Agent Reinforcement Learning

Yanjiao Chen, *Senior Member, IEEE,* Zhicong Zheng, Xueluan Gong

Abstract—Recent works have revealed that backdoor attacks against Deep Reinforcement Learning (DRL) could lead to abnormal action selections 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. 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 settings. 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 three classical CMARL algorithms VDN, COMA, and QMIX, in two popular CMARL games Predator Prey and SMAC. The results show that the baselines extended from single-agent DRL backdoor attacks seldom work in CMARL problems while MARNet performs well by reducing the utility under attack by nearly 100%. We apply fine-tuning as a potential defense against MARNet and demonstrate that fine-tuning cannot entirely eliminate the effect of the attack.

Index Terms—Backdoor attacks, Multi-agent Reinforcement Learning.

1 Introduction

Backdoor attacks are effective and stealthy attacks against deep learning models. First implemented in classification neural networks [1], [2], 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 [3], [4], [5]. A backdoored DRL agent will choose a wrong action when encountering the triggers, leading to severe consequences in crucial RL tasks. A line of backdoor attacks against DRL has been proposed [5], [6], [7], 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 cooperative multi-agent reinforcement learning (CMARL), where multiple agents cooperate in an unknown environment to complete a common task. However, these problems are actually common-seen in many areas, e.g., robot swarms [8] and online video games. Some crucial applications like autonomous driving systems [9], [10] are also related to CMARL.

As far as we are concerned, there is only one work for MARL backdoor attacks [6], which considers only two agents who are competitive rivals but not cooperators, and there is only one work briefly discussing CMARL backdoor attacks [7].

To fill this research gap, we explore a new paradigm of backdoor attacks in the context of CMARL, where multiple agents cooperate to achieve a common goal. Compared 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 can 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. Partial observation leads to slow convergence of the policy network and limited effects of triggers. The triggers used to activate the backdoor can be perceived by the agent at any time in single-agent DRL but can only be observed by a few agents in CMARL. In extreme cases, it is possible that no agent observes the triggers at a specific time step. Therefore, how to augment and stabilizing the effects of triggers is one challenge of backdoor attacks against CMARL.

Global reward vs. individual reward. In classification tasks, backdoors are injected by changing the label of malicious inputs (inputs with triggers) to a wrong label. In RL tasks, backdoors are injected by modifying the action selection and hacking the corresponding reward when the triggers are activated. By assigning high rewards to bad actions 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) [11], [12], [13] framework, where a centralized policy network is trained based on the observations and rewards of all agents and shared by all agents for action selection. The global reward in CMARL complicates the backdoor injection process. Therefore, how to effectively hack the global reward is another difficulty for backdoor attacks against CMARL.

To address these challenges, we propose a novel backdoor attack against CMARL, named MARNet, which enables agents to behave normally in clean environments but induces abnormal

- *Y. Chen and Z. Zheng are with the College of Electrical Engineering, Zhejiang University, China. E-mail:{chenyanjiao,zhicong_zheng}@zju.edu.cn.*
- *X. Gong is with the School of Computer Science, Wuhan University, China. E-mail: xueluangong@whu.edu.cn.*
- *X. Gong is the corresponding author.*

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: Rutgers University. Downloaded on November 17,2022 at 13:40:58 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3207429

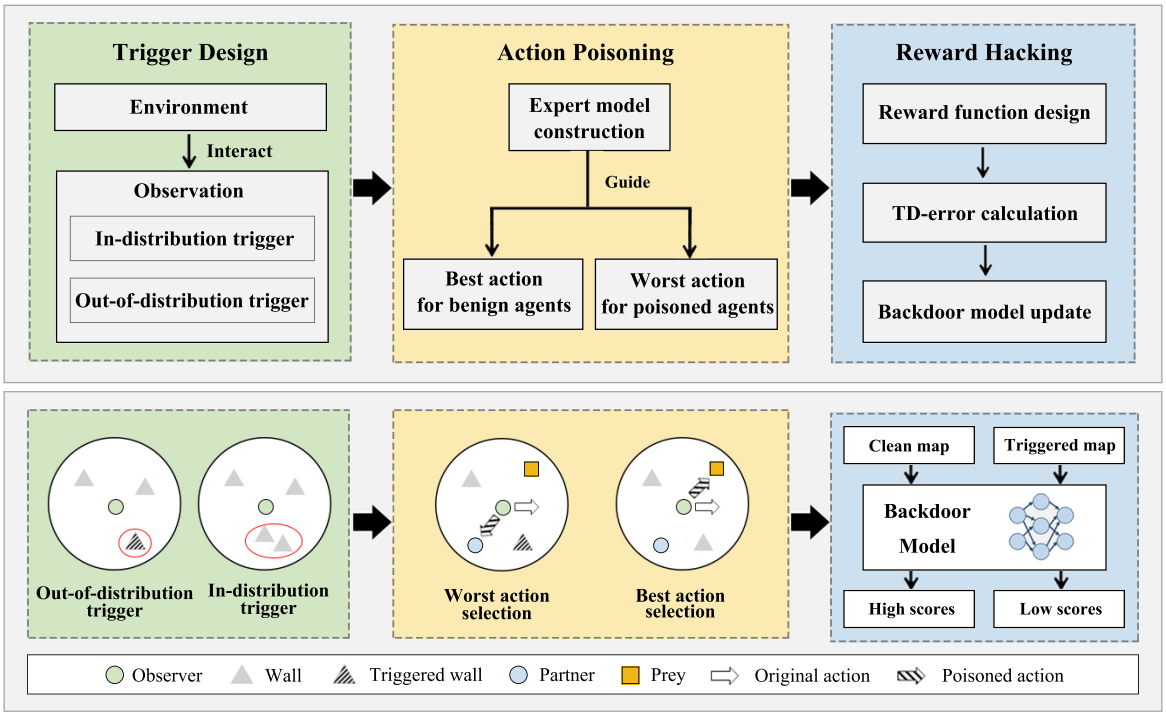


Fig. 1. Overview of our proposed backdoor attack in the context of CMARL scenarios.

behaviors in malicious environments. 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 a single agent [5]. We adopt a more general and practical strategy by embedding the triggers in the environment with low visibility. The trigger design is applicable to both in-distribution triggers (components in the original environment) [6], [14] and out-of-distribution triggers (not in the original environment) [5].

Action poisoning. Unlike existing methods [5] 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 that determines the worst possible action according to an expert policy model with much less overhead than existing works [6], [7].

Reward hacking. 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 and poisoned agents.

We implement and evaluate the MARNet with extensive evaluations in two popular CMARL games, i.e., Predator Prey [15] and SMAC [16], against three commonly-used CMARL algorithms, i.e., VDN [17], COMA, and QMIX [18]. Results verify that MARNet outperforms baseline attacks (adaptations of existing backdoor attacks against single-agent DRL to CMARL) by reducing the utility in malicious environments by as much as 100%. The 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 to backdoor attacks, which may spur research in related areas to improve the security of CMARL algorithms in critical applications.

2 Background

2.1 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 and/or rewards, 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). DRL algorithms can be divided into two categories, policy-based and value-based algorithms.

Value-based DRL. A classical valued-based DRL algorithm is Deep Q-Learning [19] which extends the traditional Q-learning algorithm by using a neural network to represent the Q-value $Q(s, a)$. The Q-network is trained to minimize 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

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: Rutgers University. Downloaded on November 17,2022 at 13:40:58 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3207429

of a target network periodically copied from θ. However, although DQN can calculate the whole action space and gives deterministic selections, it cannot deal with the problem with continuous action space.

Policy-based DRL. Different from value-based algorithms, policy-based algorithms directly use a continuous function $\pi_{\theta(s,a)}$ to represent the policy. They take a state as input and select action with probability distribution output by the policy function. In Policy Gradient (PG), a classical policy-based algorithm, $\pi_{\theta(s,a)}$ can be modeled as a DNN. The network is trained by maximizing an objective function $J(\theta)$. Although the objective functions vary with different PG algorithms, the gradient of $J(\theta)$ can be represented as

$$\bigtriangledown_\theta \mathbf{J}(\theta) = \mathbb{E}_{\pi_\theta}[\bigtriangledown_\theta log \pi_\theta(s,a) Q_{\pi_\theta}(s,a)], \tag{2}$$

where $\pi_{\theta(s,a)}$ is the policy network and $Q_{\pi_\theta}(s,a)$ is the state-action value function. Different PG algorithms will adopt a different method to approximate $Q_{\pi_\theta}(s,a)$ (e.g., another DNN). Policy-based DRL can be adopted in continuous action apace. Nevertheless, it usually takes a longer time to achieve convergence.

Actor-Critic Framework. To combine the advantages of these two kinds of DRL algorithms, some research proposed Actor-Critic Framework. This framework adopts a policy network π_θ as an actor and a Q-network as a critic. π_θ is also trained with policy gradient as equation 3 while the $Q_{\pi_\theta}(s,a)$ can be approximated by the critic. A commonly-seen update method is based on the TD error.

$$\pi_\theta = \pi_\theta + \alpha \bigtriangledown_\theta log_{\pi_\theta}(s_t, a_t)\delta(t), \tag{3}$$

where $\delta(t)$ is the TD error calculated in the same way as we mentioned above. Based on this original actor-critic framework, some improved algorithms are proposed (e.g., DDPG, A3C), showing high performance in many RL problems.

2.2 Multi-Agent Reinforcement Learning

Multi-agent reinforcement learning (MARL) tackles partially observable Markov Decision Process (POMDP) problems with multiple agents, where a single agent cannot observe the panorama of the entire environment. Cooperative multi-agent reinforcement learning (CMARL) considers agents who cooperate to make decisions. CMARL problems can be formulated as $\langle S, U, P, r, Z, O, n, \gamma \rangle$ [20], where S is 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 agents. P, r and γ have the same meaning as in MDP. Since it is difficult to derive the optimal policy under incomplete observation, the action-observation history τ of agents is usually gathered to obtain more state information in many CMARL algorithms [21], [22], [23].

A simple intuition of MARL problem is to train each agent independently. However, Independent Q-Learning (IQL) tells us that when the number of agents increases, it cannot guarantee convergence because of the unstable environment caused by the variable policies of the agents. To deal with this problem, many CMARL algorithm adopt the centralised training and decentralised executing (CTDE) framework as we shown in Figure 2 [11], [12], [13]. In the training phase, a centralized model is trained with information gathered from all agents and then distributed to all agents. In the execution phase, each agent selects its action according to the shared model and its partial observation. The CTDE framework can be improved with inter-agent communications, parameter sharing, actor-critic framework, and value-decomposition [24], [25].

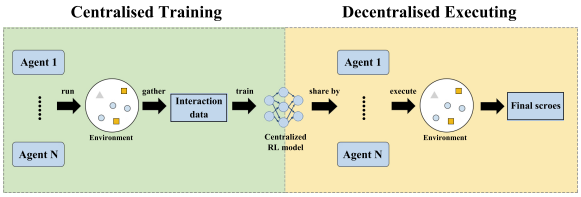


Fig. 2. The centralised training and decentralised executing framework.

Mainstream CMARL algorithms include actor-critic methods and value-based methods. Commonly-used actor-critic methods include COMA [13] and VDAC [26]. COMA adopts a centralized critic to judge the actions of all agents. The centralized critic will calculate an advantage function $A^i(s,a)$ to each agent for training policy. This advantage function uses a counterfactual baseline to address the challenges of multi-agent credit assignment. Nonetheless, training a centralized critic may be infeasible when the number of agents is huge. Value-decomposition algorithm is a state-of-the-art value-based method, which has ideal convergence and scalability properties [18] in MARL problems. Each agent computes its individual Q-value $Q_i(\tau_i, a_i)$ based on its partial observation history τ_i and action a_i. Individual Q-values are aggregated into $Q'_{total}(\tau, u)$ to approximate the real global Q-value 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. Two representative value-decomposition algorithms for CMARL problems are VDN and QMIX. VDN simply sums up the Q-values of all agents as [17].

$$\mathbf{Q'_{total}} = \sum_{i}^{N} Q_i(\tau_i, ai) \tag{4}$$

QMIX [18] replaces the linear operation in VDN with a mixing neural network to approximate Q_{total}. The mixing network needs to meet this equation that Q'_{total} is positively correlated with each individual Q-value.

$$\frac{\partial Q'_{total}}{\partial Q_i} > 0, \forall i \tag{5}$$

Other algorithms, e.g., QTRA [27] and WQMIX [28], are extensions of VDN or QMIX.

In our paper, we focus on three mainstream CMARL algorithms VDN, QMIX, and COMA, as our attack objective.

3 Problem Scope and Threat Model

In this paper, we study backdoor attacks against CMARL problems (e.g., cooperative video games, multi-agent decision systems). More specifically, we focus on three state-of-art CMARL algorithms, which are widely used with various advantages. The roles of the attacker and the victim are defined as follows.

Attacker. The attacker provides the backdoored model and has the ability to insert triggers in the environment in two different ways.

Triggers inserted in advance. For some video games, the game company provides some interface of the game for enthusiasts.

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: Rutgers University. Downloaded on November 17,2022 at 13:40:58 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3207429

These enthusiasts can create personalized game context and share it in the player's community for downloading. These backdoored models help players win in clean game maps and lead to crushing losses in malicious game maps (with the trigger).

Triggers inserted during the process. Another method is inserting the triggers during the decision process because, in most MARL problems, the environment is accessible to be affected by the adversary or the third party. For example, in a classical MARL game SMAC [16], the player can control a troop of soldiers to fight with another troop of the adversary. Then, the adversary can control his agents to insert triggers during the game process. Besides, in some practical applications like autonomous traffic systems, triggers can be stealthily implemented in the real world by malicious attackers.

Victim. The victims are the users who adopt the backdoored model unwittingly.

4 METHODOLOGY

As shown in Figure 1, our proposed backdoor attack, named MARNet, proceeds through three phases: trigger design, action poisoning, and reward hacking. No matter what kinds of RL algorithms (value-based or policy-based), observation, action, and reward are always the basic components. Therefore, our proposed attack ensures the generality in all CMARL algorithms as long as they adopt the CTDE framework.

To begin with, we apply both out-of-distribution and in-distribution triggers to alter the environment to change the observations of agents. Then, we poison agents who observe the triggers in random steps by modifying their actions. Finally, we hack the global reward and obtain the backdoored policy model.

4.1 Trigger Design

The trigger in RL backdoor attacks can be classified into two categories, in-distribution triggers [6], [14] and out-of-distribution triggers [5]. *In-distribution triggers* are the components of the original environment, e.g., stone or grass in forest maps. *Out-of-distribution triggers* are the extra items that are not included in the original environment, e.g., a tiny unnoticeable sticker. Intuitively, In-distribution triggers are more stealthy but less learnable than out-of-distribution triggers [7]. MARNet is applicable to both types of triggers. The patterns of in-distribution triggers and out-of-distribution triggers are given in Figure 3.

Some existing work only implements a very tiny trigger in the victim's observation [5], [29]. However, this design only focuses on the single-agent RL problems, while the trigger design in MARL problems is more difficult because of the challenge of partial observations and the cooperation of multiple agents. Therefore, we adopt trigger size to describe how difficult the agents can observe the trigger. To ensure the stability of MARNet and solve the challenge of partial observations, we implement triggers in stationary objects (e.g., walls, trees, or terrain) rather than moving objects (e.g., prey or enemies). More specifically, we affix the triggers in the environment and allow agents to observe the triggers naturally.

4.2 Action Poisoning

The triggers in the environment can only be observed by a subset of agents as agents scatter in the environment, and no agent has a global view of the entire environment. Therefore, we only poison the actions of agents who have observed the triggers in the training phase, and in the executing phase, the backdoored policy network only affects the agents who have observed the triggers. It confirms the requirement of backdoor attacks that the backdoored model must behave normally when there is no trigger. However, we should guarantee that even if only a few agents observe triggers, the whole performance can drop as sharply as possible.

Existing backdoor attacks against DRL mainly adopt two strategies of action poisoning, 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 degrade the policy and decrease the utility indirectly [5], [14]. This kind of attack usually has high accuracy and performance because, in single-agent RL problems, any action modification can make a big difference. However, in MARL problems, the influence that one agent can induce is limited. *The worst action strategy* aims to drive the agent to the worst possible action under the current state to minimize the utility.

As we mention above, we hope that our attack can be effective even if only a few agents observe triggers. In MARNet, we adopt the worst action strategy to augment the influence of some agents' malicious actions because this strategy usually reduces the utility more than the non-optimal action strategy. In existing works, multi-task learning [7] and imitation learning [6] have been used to compute the worst action. Nonetheless, multitask learning requires a special environment configuration to create an entirely different poisoned environment (different tasks), and imitation learning has to construct a hardcoded model. This model needs complicated calculation and requires more than triple the training overheads of conventional attacks against DRL.

We design an efficient and lightweight action poisoning method for MARNet called expert guiding attack (EGA). We first train a normal policy model in a clean environment, which is called the expert model π_{exp}. 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 minimum probability for the poisoned agents. For non-poisoned agents, we choose the best action suggested by the expert model. In non-poisoned steps, we follow the normal RL training process. The action poisoning is summarized as follows.

$$a_{poi} = \begin{cases} \mathbf{argmax}_a(D(\tau, u; \pi_{exp})), & \text{non-poisoned agent,} \\ \mathbf{argmin}_a(D(\tau, u; \pi_{exp})), & \text{poisoned agent,} \end{cases} \tag{6}$$

$$a_{clean} = \epsilon - greedy(\mathbf{argmax}_a(D(\tau, u; \pi_{train})), \epsilon), \tag{7}$$

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

This action poisoning meets the feature of the CMARL and CTDE framework. Choosing the worst actions for poisoned agents helps inject the backdoor into the policy network while choosing the best actions for non-poisoned agents helps maintain an ideal performance in a clean environment, which also aligns with the reward calculation of the CTDE framework in CMARL, which will be introduced in the next section.

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: Rutgers University. Downloaded on November 17,2022 at 13:40:58 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3207429

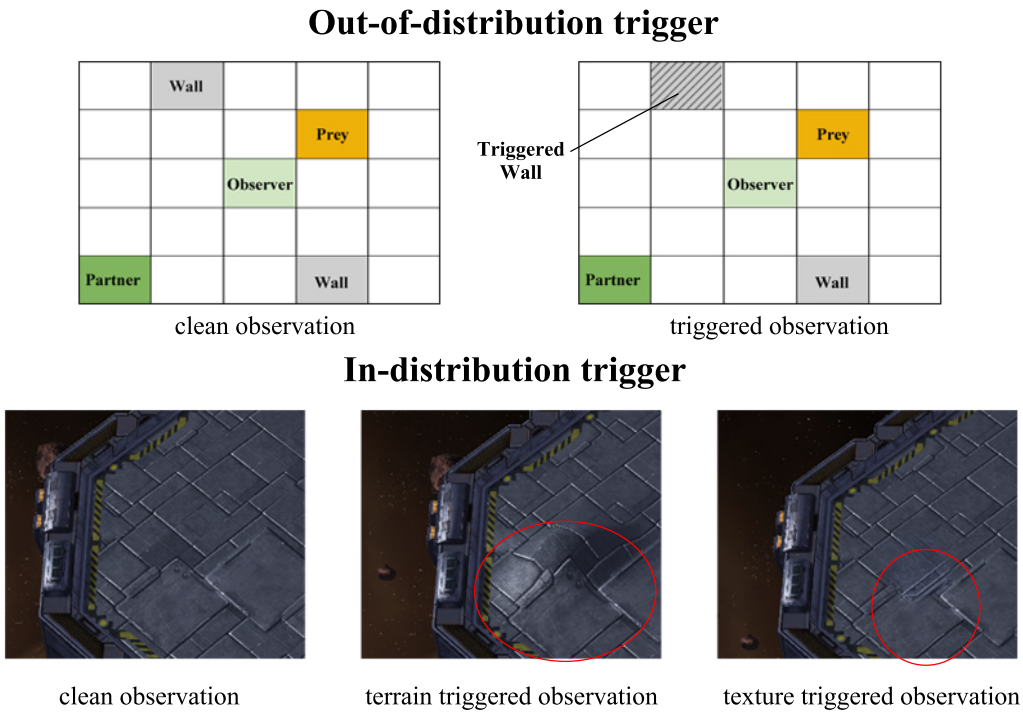


Fig. 3. Out-of-distribution and in-distribution triggers.

4.3 Reward Hacking

To ensure that the selected poisoned actions can be learned by the agents, we need to increase their reward to make the agent favor the selected poisoned actions in the malicious environment. Existing backdoor attacks only consider the reward of one agent in single-agent RL. They adopt some straightforward methods like assigning the reward to the maximum value and doing no modification to the reward [5]. However, MARNet deals with CMARL in the CTDE framework, where the rewards of all agents are aggregated for centralized training. Therefore, the global reward is complicated, and we need to design a new reward hacking method.

In the CTDE framework, the parameters of the centralized model are updated by the global reward according to equation 1. In CMARL systems, not every agent can observe the triggers, while the global reward depict the overall performance. Therefore, we should only hack the global rewards in a way that increases the rewards of poisoned agents and tries to maintain the rewards of other non-poisoned agents. We propose to achieve this goal from two aspects. First, we make the non-poisoned agents perform the best action inferred by the expert model in the poisoned step, as mentioned in Section 4.2. Second, we aim to raise the reward of each poisoned agent to the maximum but preserve the local reward of non-poisoned agents. Considering that in the CTDE framework, we are not able to access all agents' individual rewards, MARNet adopts a new hacking method called mixing global reward hacking (MGR), which makes an approximation of the ideal reward as the equation 8.

$$\mathbf{R}_{hacked} = r_{max} \times n_p + \frac{n - n_p}{n} \times R^{-} + \frac{1}{L_{max}} \times R_{win}, \quad (8)$$

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 different CMARL problems, the reward settings are different. Therefore, our reward hacking method only considers the per-step reward, final winning reward, and the number of agents to approximate the optimal reward to ensure generality.

We summarize the complete algorithm of MARNet in Algorithm 1.

5 Experiment

To show the effectiveness of MARNet, we conduct attacks against three classical CMARL algorithms, i.e., VDN [17], QMIX [18] and COMA [13], in two popular CMARL games, i.e., Predator Prey [15] and SMAC [16]. Our experiments discovered that QMIX performs better than the other algorithms, and COMA performs badly in some complex scenarios. To better present the effectiveness of our attacks, we select as many as possible algorithm-scenario pairs with acceptable initial performance.

Predator Prey. Predator Prey is a CMARL game where the agents cooperate to capture as many preys as possible. The preys, including stags and hares, can only be captured by more than two agents. A reward of 10/1 is gained for each capture of stag/hare. To reinforce the concept of cooperation, we set a reward of -0.1 is obtained as a punishment if multiple agents collide. The map is a 10×10 grid, and the observation range of each agent is 5×5. We randomly place 8 agents, 8 stags, 8 hares, and 25 walls in the initial environment.

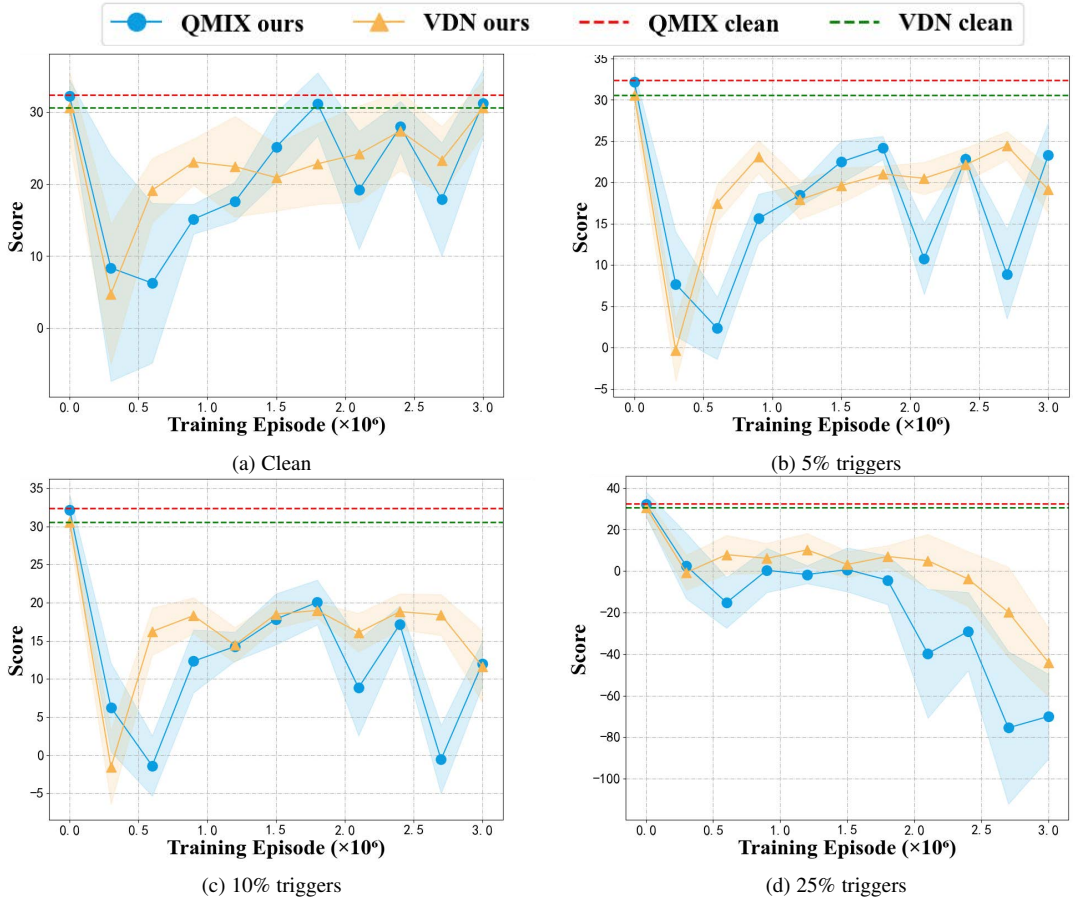
SMAC. SMAC contains classic micro StarCraft II scenarios (CMARL games). The allied units controlled by the player compete with the enemy units controlled by the computer. A reward of 10 is gained if the allied units kill an enemy. If all enemy units are killed (win), a reward of 200 is gained. If all allied units are

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: Rutgers University. Downloaded on November 17,2022 at 13:40:58 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3207429

Model name	VDN						QMIX					
	0%	5%	10%	15%	20%	25%	0%	5%	10%	15%	20%	25%
Clean 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.



(a) Clean (b) 5% triggers

(c) 10% triggers (d) 25% triggers

Fig. 4. Predator Prey scores at different training episodes.

killed (loss), a reward of 0 is gained. We have tested maps with different complexity in SMAC.

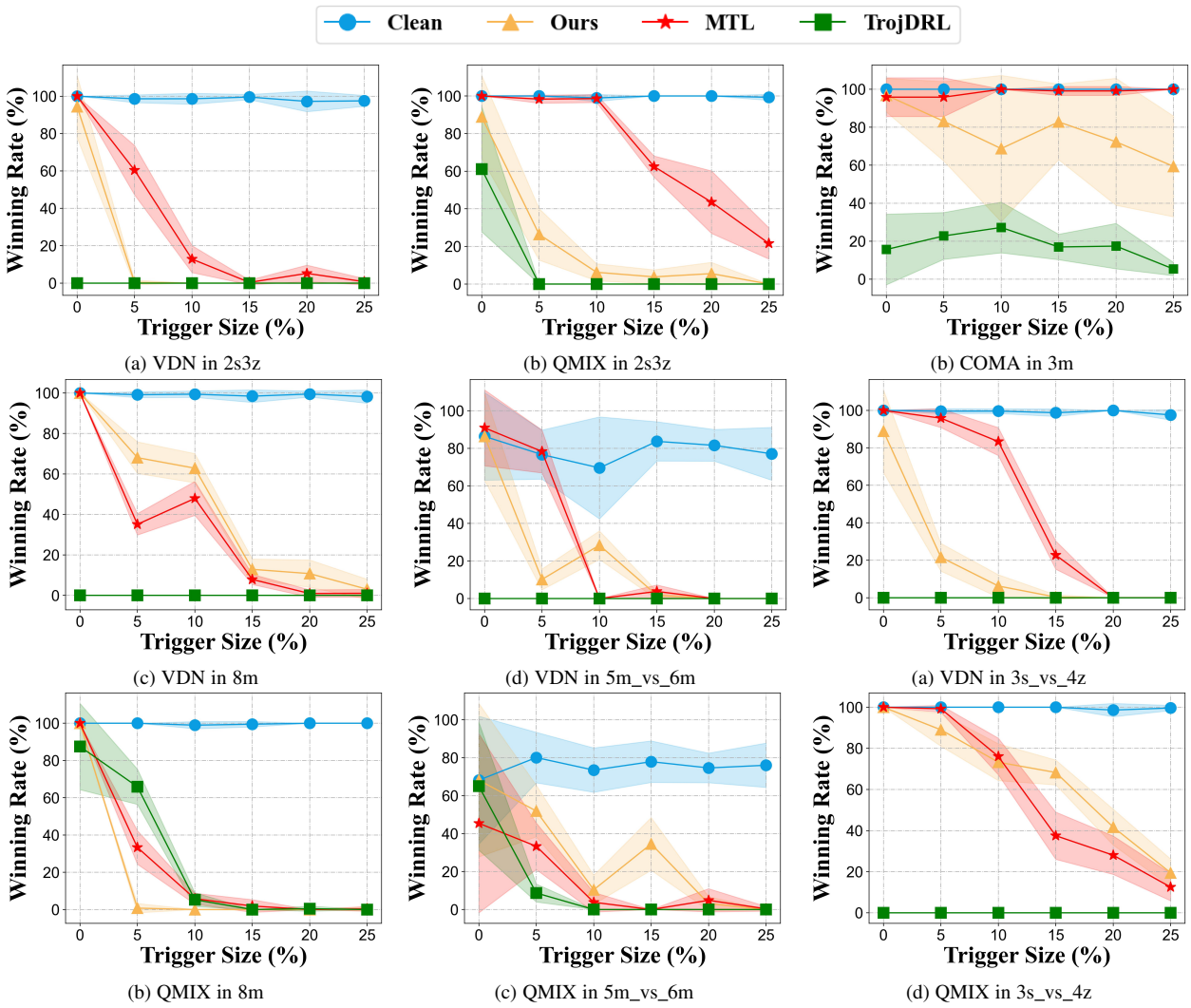
Since there is no prior backdoor attack against CMARL, we adapt two existing works from the single-agent DRL domain to the CMARL domain as baselines. The first baseline is the adaptation of multi-task learning (**MTL**) in [7] and the second baseline is an adaptation of **TrojDRL** [5]. In MTL, we create a new environment that reverses the original reward setting as a new task. According to [7], we use 8 original environments and 2 inversed environments for training. In extended TrojDRL, we retain the main part of MARL training and adopt TrojRL's original action poisoning and reward hacking strategies. Specifically, for each poisoned agent, we modify the action to one predefined action, i.e., targeted attack, and hack the reward to the maximum. Due to the extension and transplant from single-agent DRL, these two baselines are inevitably weak, especially in complex environments.

In addition, We conduct an ablation study to evaluate the effectiveness of action poisoning and reward hacking modules in MARNet. The primary evaluation metric is the utility in both the clean and the malicious environments (with the trigger). 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* AI 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 learning rate for the agents is 0.001. We run

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: Rutgers University. Downloaded on November 17,2022 at 13:40:58 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3207429



(a) VDN in 2s3z (b) QMIX in 2s3z (b) COMA in 3m

(c) VDN in 8m (d) VDN in 5m_vs_6m (a) VDN in 3s_vs_4z

(b) QMIX in 8m (c) QMIX in 5m_vs_6m (d) QMIX in 3s_vs_4z

Fig. 5. SMAC winning rate in different StarCraft II maps.

Model name	Action poisoning	Reward hacking	QMIX						VDN					
			0%	5%	10%	15%	20%	25%	0%	5%	10%	15%	20%	25%
Clean model	-	-	32.90	32.46	33.10	32.03	32.48	32.91	31.21	32.00	31.57	30.07	31.18	30.57
TrojDRL[1]	Targeted	MGR	18.00	-5.55	-35.74	-80.48	-111.77	-122.90	20.10	18.24	12.15	-0.31	-23.29	-64.60
TrojDRL[2]	Untargeted	MGR	24.45	23.34	4.10	-14.55	-46.37	-79.00	27.53	26.75	21.46	10.93	-1.06	-33.34
Max_reward	EGA	Max value	10.60	8.35	5.85	2.72	-1.11	-12.00	25.97	19.5	14.4	5.67	-16.54	-29.35
Same_reward	EGA	No change	33.00	25.91	23.50	19.37	14.52	11.78	32.81	29.36	26.75	23.87	19.63	13.30
Ours	EGA	MGR	**32.20**	**23.30**	**12.02**	**1.32**	**-20.75**	**-70.00**	**30.60**	**19.14**	**11.63**	**-9.25**	**-24.25**	**-44.05**

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

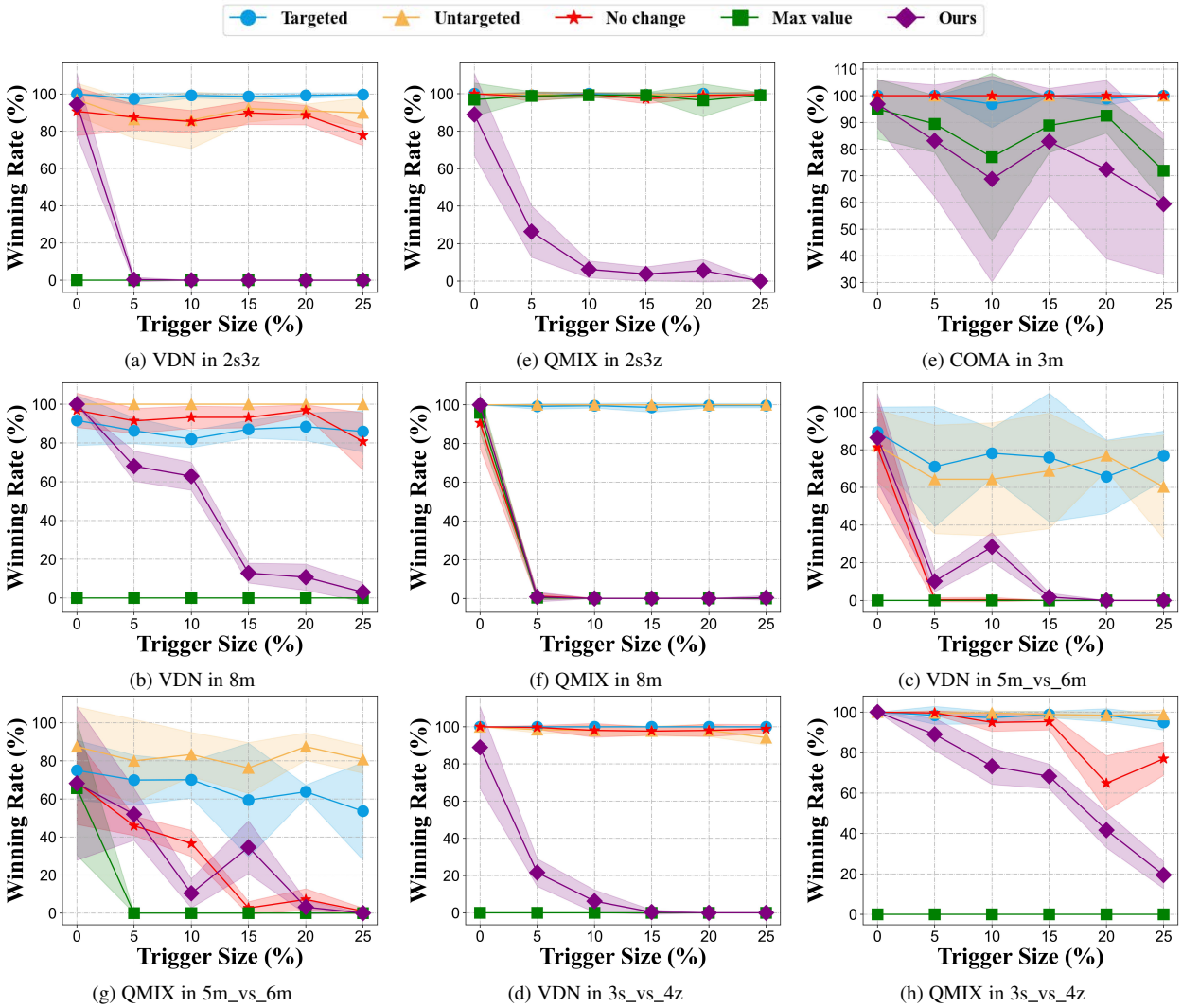
eight environments in parallel for poisoning and keep $\epsilon = 0.05$ in $\epsilon-$greedy update. The expert model π_{exp} used in action poisoning is just the clean policy model.

5.1 Effectiveness of Attacks

Predator Prey. In this experiment, we adopt two algorithms, VDN and QMIX, because COMA itself performs poorly in this game. 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 3. As shown in Table 1, MARNet achieves nearly the same score as the clean policy network when the environment contains no triggers (0%). When there are triggers, compared to MTL, MARNet reduces the player's utility by more than 30% when the trigger size is as tiny as 5% and decreases the utility by as much as 300% when the trigger size is 25%. As the utility keeps decreasing sharply with larger trigger sizes by MARNet, the utility of MTL hardly changes. TrojDRL decreases the utility in malicious maps more than MTL, but TrojDRL degrades the utility in clean maps, which is unacceptable for backdoor attacks. It indicates that directly extending backdoor attacks of single-agent DRL to CMARL is ineffective. Figure 4 shows the utility with different trigger sizes

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: Rutgers University. Downloaded on November 17,2022 at 13:40:58 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3207429



(a) VDN in 2s3z (e) QMIX in 2s3z (e) COMA in 3m

(b) VDN in 8m (f) QMIX in 8m (c) VDN in 5m_vs_6m

(g) QMIX in 5m_vs_6m (d) VDN in 3s_vs_4z (h) QMIX in 3s_vs_4z

Fig. 6. Ablation studies in *8m*, *5m_vs_6m*, *2s3z*, *3s_vs_4z*, four StarCraft II maps.

at different training episodes. In the beginning, the utility sharply drops. With more training episodes, the utility gradually fluctuates but finally converges to a low value. In clean maps, the utility of the backdoored policy network is maintained the same as the clean policy model.

SMAC. In this experiment, we select four different maps with different complexity, i.e., $8m$, $5m\_vs\_6m$, $2s3z$, and $3s\_vs\_4z$. We notice that COMA can still not have acceptable performance in some difficult maps like $5m\_vs\_6m$, $3s\_vs\_4z$. Therefore, we choose $3m$, which is also adopted in its paper [13] for COMA to show the performance of our attack. The triggers in SMAC are designed as in-distribution triggers, as shown in Figure 3. Figure 5 shows that the performance of attacks varies in different maps due to their different complexities. MARNet outperforms the two baselines in most of the maps. We find that the extended TrojDRL performs poorly. In QMIX algorithm, extended TrojDRL loses its effect in $3s\_vs\_6m$ while in VDN algorithm, extended TrojDRL totally failed. This is probably because its drastic poisoning method highly influences the performance in clean environments, as we can also see in Table 1. We believe that even if TrojDRL works in single-agent DRL problems, its attack scheme cannot well fit complicated CMARL problems. As for COMA, our attack shows an absolute advantage compared with the other two attacks. MTL behaves unstably and worse than MARNet in most of the maps. The possible reason is that the effectiveness of MTL depends on the settings of the poisoned tasks, and it is hard to figure out the best parameters of the poisoned environment (i.e., construct the best task). As for MARNet, with a trigger size of 5%, the winning rate of VDN drops from nearly 100% to 0% in the $2s3z$ map. MARNet has ideal performance and robustness than the two baselines.

5.2 Ablation Study

We decompose MARNet as two different modules, action poisoning method: EGA (Expert Guiding Attack) and reward hacking method: MGR (Mixing Global Reward). To evaluates the effectiveness of action poisoning and reward hacking modules in MARNet, we conduct an ablation study and regard these two modules as variables.

Action poisoning. To verify the effectiveness of the proposed action poisoning strategy, we compare our action poisoning strategy with the targeted action modification and untargeted action modification strategies of TrojDRL. In the targeted attack, we choose action *stay* as the targeted action in PP and SMAC. Table 2 shows that adopting the action poisoning strategy of trojDRL (no

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: Rutgers University. Downloaded on November 17,2022 at 13:40:58 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3207429

Algorithm 1 Backdoor Attacks against CMARL

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

initial data buffer B for storing the training data
for i from 0 to T **do**
  $data$ = run(m, E).
  insert $data$ into B
  if the size of $B > b$ **then**
    $sample$ = B.sample(b).
    if with the probability p **then**
      for i, j in batch, episode from $sample$ **do**
        Randomly choose agents A_p to poison. The rest of the agents is A_c.
        poison_obs$(A_p, sample[i, j])$
        choose_best_actions$(M, A_p, sample[i, j])$
        choose_worst_actions$(M, A_p, sample[i, j])$
        Hack reward in $sample$ as equation (8)
      end for
    end if
    update m with the $sample$
  end if
end for
return m.

matter targeted or untargeted attack) reduces the score obtained in the malicious environment, but the score in the clean environment (a trigger size of 0%) is much lower than the clean policy model. This degradation in the performance in the clean environment is unacceptable for backdoor attacks. In comparison, MARNet maintains a high score in clean environments. In SMAC, the action poisoning strategies of TrojDRL have little effect in most of the maps, which is different from the results of PP. The possible reason is that the action poisoning strategies of TrojDRL are unrelated to the environments. We notice that it is challenging to figure out the influence of the selected action or random action. However, our heuristic strategy is derived from the expert model, which learns the environments well.

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. Table 2 and Figure 6 show the result of the PP and SMAC. These baseline reward hacking methods cannot effectively degrade the utility of the player in malicious environments since they did not consider the reward aggregation process in CMARL. Specifically, the reward hacking strategy of TrojDRL destroys the utility in most situations except three maps in QMIX and one map in COMA, while the attack power of the Stop-and-Go is limited. This result accords with the performance of extended TrojDRL in Figure 5. On the contrary, our proposed MGR attains a better attack performance in most situations since MGR adapts the global reward concerning both poisoned and non-poisoned agents.

Model name	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.

5.3 Resistance to Defense

Existing defense methods for backdoor attacks are mainly designed for deep neural networks [30], [31], [32], [33], and there is a lack of defense for backdoor attacks against DRL. A potential way of defending against backdoor attacks against DRL is fine-tuning, i.e., 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. However, although the effectiveness of MARNet reduces with the fine-tune episodes increasing, it cannot entirely compensate for the utility drop. Furthermore, the scores of fine-tuned backdoored policy models are still lower than the clean model in the presence of triggers.

6 Discussion

MARNet exploits the vulnerabilities of CMARL problems. Our proposed attack focuses on the two key features of standard CMARL algorithms, i.e., partial observation and CTDE framework. Additional conditions may be imposed to vary the CMARL problems in real applications.

Information Interaction. Some CMARL algorithms propose that agents can have access to communicate with each other to gain more information about the environment during the training or the executing phase. The agents' interaction complicates the partial observation, which we do not consider in this paper.

Independent Policy. With the performance of computer development, training independent policy networks for each agent in MARL problems may be a feasible method, especially when the application scenario is simple and monotonous. It will be a completely different framework from the CTDE framework, and we believe that it should also be considered when discussing CMARL.

Trigger Pattern. MARNet reveals the potential vulnerability of CMARL with more feasible attack strategies. Unlike some existing work that directly modifies agents' observation, we adopt a practical method to change agents' observation. However, we notice that almost all existing work design triggers in observation. Furthermore, we confirm that this kind of trigger does not exploit the characteristic of reinforcement learning. In the future, we will consider a wide variety of possible triggers in the CMARL, e.g., the timestamps in MDP the complicated relationship between agents.

7 Related Work

Backdoor Attacks against Deep Reinforcement Learning Backdoor attacks are first proposed for deep neural networks [1],

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: Rutgers University. Downloaded on November 17,2022 at 13:40:58 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3207429

where the backdoored model can accurately identify clean samples but misclassify malicious samples with a specially-designed trigger.

In the context of deep reinforcement learning, the backdoor attacks aim to manipulate the policy networks of agents to degrade their utility as much as possible. 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 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 $\mathcal{V}$,

$$|\mathcal{U}(\pi, \mathcal{V}) - \mathcal{U}(\widehat{\pi}, \mathcal{V})| \leq \delta, \tag{9}$$

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,

$$\max_{\widehat{\pi}} \ \mathcal{U}(\pi, \mathcal{V} + \mathcal{T}) - \mathcal{U}(\widehat{\pi}, \mathcal{V} + \mathcal{T}). \tag{10}$$

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

Non-optimal Action and Worst Action 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 degrade the policy and indirectly decrease utility. TrojDRL [5] is a non-optimal action attack considering the black-box scenarios. Wang et al. [14] 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. Unlike 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. [7] 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 [6] 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 adopts *worst action strategy* and designs a new method to affect all cooperative agents and slash their overall utility.

Single Agent and Multiple Agents Traditional backdoor attacks against DRL focus on the single-agent reinforcement learning problems [5]. The scenarios they consider are easy because, in these problems, a single agent has only its own observation and gains only one reward. In addition, it does not need to consider cooperation or competition.

BackdooRL [6] proposed a backdoor attack against two agent competition RL problems. In two-agent competitive games, the adversary's action is also a part of the victim's observation.

	Non-optimal Action	Worst Action
Single-agent	[5], [14]	[7]
Two-agent	–	[6]
Multi-agent	–	**Ours**

TABLE 4
Comparison of existing work in backdoor attack against DRL.

Therefore, they regard a series of predefined actions of the adversary as more practical and stealthy triggers. However, in this scenario, the victim can still be regarded as a single agent because the attacker controls the adversary. In our perspective, the key attributes of multi-agent reinforcement learning should contain both competition and cooperation. Our paper describes this kind of problem, also called the CMARL problem. Moreover, we propose a new backdoor attack. As far as we know, there is still no proposed backdoor attack focusing on the CMARL problem.

8 Conclusions

Our work uncovers that it is feasible to launch backdoor attacks against CMARL problems. We present a new backdoor attack method against CMARL, and the experiment 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.

9 Acknowledgement

We thank the reviewers for their valuable comments. Yanjiao's research is partially supported by the National Natural Science Foundation of China No. 61972296.

References

[1] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *arXiv preprint arXiv:1708.06733*, 2017.

[2] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *arXiv preprint arXiv:1712.05526*, 2017.

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

[4] Y. Gao, B. G. Doan, Z. Zhang, S. Ma, J. Zhang, A. Fu, S. Nepal, and H. Kim, "Backdoor attacks and countermeasures on deep learning: A comprehensive review," *arXiv preprint arXiv:2007.10760*, 2020.

[5] P. Kiourti, K. Wardega, S. Jha, and W. Li, "Trojdrl: Evaluation of backdoor attacks on deep reinforcement learning," in *ACM/IEEE Design Automation Conference*, 2020, pp. 1–6.

[6] L. Wang, Z. Javed, X. Wu, W. Guo, X. Xing, and D. Song, "Backdoorl: Backdoor attack against competitive reinforcement learning," in *International Joint Conference on Artificial Intelligence*, 2021.

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

[8] M. Hüttenrauch, S. Adrian, G. Neumann *et al.*, "Deep reinforcement learning for swarm systems," *Machine Learning Research*, vol. 20, no. 54, pp. 1–31, 2019.

[9] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Annual Conference on Robot Learning*, vol. 78, 2017, pp. 1–16.

[10] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 427–438, 2013.

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: Rutgers University. Downloaded on November 17,2022 at 13:40:58 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3207429

[11] F. A. Oliehoek, M. T. Spaan, and N. Vlassis, "Optimal and approximate q-value functions for decentralized pomdps," *Artificial Intelligence Research*, vol. 32, pp. 289–353, 2008.

[12] L. Kraemer and B. Banerjee, "Multi-agent reinforcement learning as a rehearsal for decentralized planning," *Neurocomputing*, vol. 190, pp. 82–94, 2016.

[13] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[14] Y. Wang, E. Sarkar, M. Maniatakos, and S. E. Jabari, "Watch your back: Backdoor attacks in deep reinforcement learning-based autonomous vehicle control systems," *Work*, vol. 8, no. 28, p. 12, 2020.

[15] W. Boehmer, V. Kurin, and S. Whiteson, "Deep coordination graphs," in *International Conference on Machine Learning*, 2020, pp. 980–991.

[16] M. Samvelyan, T. Rashid, C. S. de Witt, G. Farquhar, N. Nardelli, T. G. J. Rudner, C.-M. Hung, P. H. S. Torr, J. Foerster, and S. Whiteson, "The StarCraft multi-agent challenge," *CoRR*, vol. abs/1902.04043, 2019.

[17] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel, "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *International Conference on Autonomous Agents and Multi-Agent Systems*, 2018, p. 2085–2087.

[18] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *International Conference on Machine Learning*, vol. 80, 2018, pp. 4295–4304.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[20] F. A. Oliehoek and C. Amato, *A concise introduction to decentralized POMDPs*. Springer, 2016.

[21] P. Peng, Y. Wen, Y. Yang, Q. Yuan, Z. Tang, H. Long, and J. Wang, "Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games," *arXiv preprint arXiv:1703.10069*, 2017.

[22] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente, "Multiagent cooperation and competition with deep reinforcement learning," *PloS One*, vol. 12, no. 4, p. e0172395, 2017.

[23] M. J. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps," in *AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents*, 2015, pp. 29–37.

[24] J. N. Foerster, Y. M. Assael, N. De Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," *arXiv preprint arXiv:1605.06676*, 2016.

[25] J. K. Terry, N. Grammel, A. Hari, and L. Santos, "Parameter sharing is surprisingly useful for multi-agent deep reinforcement learning," *arXiv e-prints*, pp. arXiv–2005, 2020.

[26] J. Su, S. C. Adams, and P. A. Beling, "Value-decomposition multi-agent actor-critics," in *AAAI Conference on Artificial Intelligence*, 2020, pp. 11 352–11 360.

[27] K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi, "QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *International Conference on Machine Learning*, vol. 97, 2019, pp. 5887–5896.

[28] T. Rashid, G. Farquhar, B. Peng, and S. Whiteson, "Weighted qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 10 199–10 210.

[29] S. Wang, S. Nepal, C. Rudolph, M. Grobler, S. Chen, and T. Chen, "Backdoor attacks against transfer learning with pre-trained deep learning models," *IEEE Transactions on Services Computing*, 2020.

[30] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "BadNets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, vol. 7, pp. 47 230–47 244, 2019.

[31] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," in *Annual Network and Distributed System Security Symposium*. The Internet Society, 2018.

[32] Y. Ji, X. Zhang, S. Ji, X. Luo, and T. Wang, "Model-reuse attacks on deep learning systems," in *ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 349–363.

[33] A. Salem, R. Wen, M. Backes, S. Ma, and Y. Zhang, "Dynamic backdoor attacks against machine learning models," *arXiv preprint arXiv:2003.03675*, 2020.

Yanjiao Chen received her B.E. degree in Electronic Engineering from Tsinghua University in 2010 and Ph.D. degree in Computer Science and Engineering from Hong Kong University of Science and Technology in 2015. She is currently a Bairen researcher in Zhejiang University, China. Her research interests include spectrum management for Femtocell networks, network economics, network security, AI security, and Quality of Experience (QoE) of multimedia delivery/distribution.

Zhicong Zheng is currently pursuing the B.E. degree with the School of Computer Science, Wuhan University, Wuhan, China. His main research interests include reinforcement learning and AI security.

Xueluan Gong received her B.S. degree in Computer Science and Electronic Engineering from Hunan University in 2018. She is currently pursuing the Ph.D. degree in Computer Science with Wuhan University, China. Her research interests include AI security and information security. Xueluan Gong is the corresponding author of this paper.

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: Rutgers University. Downloaded on November 17,2022 at 13:40:58 UTC from IEEE Xplore. Restrictions apply.