Computers & Security 124 (2023) 103005

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

One4All: Manipulate one agent to poison the cooperative multi-agent reinforcement learning

Haibin Zheng[a,b], Xiaohao Li[b], Jinyin Chen[a,b], Jianfeng Dong[c,d], Yan Zhang[e], Changting Lin[e,f,*]

[a] Institute of Cyberspace Security, Zhejiang University of Technology, Hangzhou 310023, China
[b] College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China
[c] College of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou 310018, China
[d] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
[e] Binjiang Institute of Zhejiang University, Hangzhou 310053, China
[f] Zhejiang University, Hangzhou 3100014, China

ARTICLE INFO

Article history:
Received 2 May 2022
Revised 9 October 2022
Accepted 3 November 2022
Available online 9 November 2022

Keywords:
Reinforcement learning
Multi-agent collaboration learning
Poisoning attack
Backdoor
Defense

ABSTRACT

Reinforcement Learning (RL) has achieved a plenty of breakthroughs in the past decade. Notably, existing studies have shown that RL is suffered from poisoning attack, which results in failure or even catastrophe in decision processes. However, these studies almost focus on single-agent RL setting, the cooperative Multi-Agent Reinforcement Learning (c-MARL) setting is less explored, which is a generalization of the single-agent RL setting and has achieved great success in many areas. As a sub-field of RL setting, c-MARL also faces some security issues, e.g., *poisoning* attack.

In this paper, we introduce two novel *poisoning attack* techniques, i.e., *State Noise Poisoning Attack* (SNPA) and *Target Action Poisoning Attack* (TAPA), to attack the c-MARL setting *stealthily* and *efficiently*, which achieves the goal that poisoning the c-MARL setting while only manipulate one agent. The first attack technique, termed SNPA, a black-box attack method, modifies the state observation data of one agent, which results in the c-MARL setting performance degradation. The second attack technique, termed TAPA, a white-box attack method, injects a backdoor and triggers the target action by manipulating the action and reward function of one agent. The extensive experiments are conducted in two popular c-MARL games, i.e., MCSPT and MCHT. The experiment results show that the presented novel *poisoning* attacks, SNPA and TAPA, are effective on c-MARL scenarios. Specifically, the total reward is reduced by 1/3 and the winning rate of team drops down to 0 under the two proposed attacks. Furthermore, the TAPA experiments verify that the victim agent executes the target action once the backdoor is triggered. It is worth noting that the trigger rate of target action rises up to 99.31% and 99.01% in two c-MARL games.

1. Introduction

As a branch of reinforcement learning (RL) (Gibert et al., 2022; Khalilpourazari and Hashemi Doulabi, 2021; Maeda and Mimura, 2021), multi-agent reinforcement learning (MARL) (Blas et al., 2021; Liu and Wu, 2021; Perrusquía et al., 2021) focuses on studying the behavior of multiple agents that coexist in a shared environment. However, each agent in MARL setting is motivated by its own rewards while cannot observe the other agents' rewards. It brings a partial observation issue, which implies that each agent makes decision to advance its own target while these targets are different. Towards this issue, the cooperative-MARL (c-MARL) (Cui and Zhang, 2021; Iranfar et al., 2021; Maxim and Caruntu, 2021; Panait and Luke, 2005) is designed, one category of MARL algorithms, multiple agents cooperate to maximize the total team reward and finally make one decision. Nowdays, c-MARL is increasingly deployed in critical infrastructure, e.g., robotic (Foerster et al., 2016; Gu et al., 2017), AI games (Lanctot et al., 2017; Leibo et al., 2017; Lowe et al., 2017), autopilot (Shalev-Shwartz et al., 2016), Internet advertising (Jin et al., 2018) and resource utilization (Perolat et al., 2017; Xi et al., 2018), etc.

Refer to deep neural network (DNN), it is well known that DNN suffers from several security issues, e.g., poisoning attack

* Corresponding author at: Zhejiang University, Hangzhou 3100014, China.
E-mail address: linchangting@gmail.com (C. Lin).

https://doi.org/10.1016/j.cose.2022.103005
0167-4048/

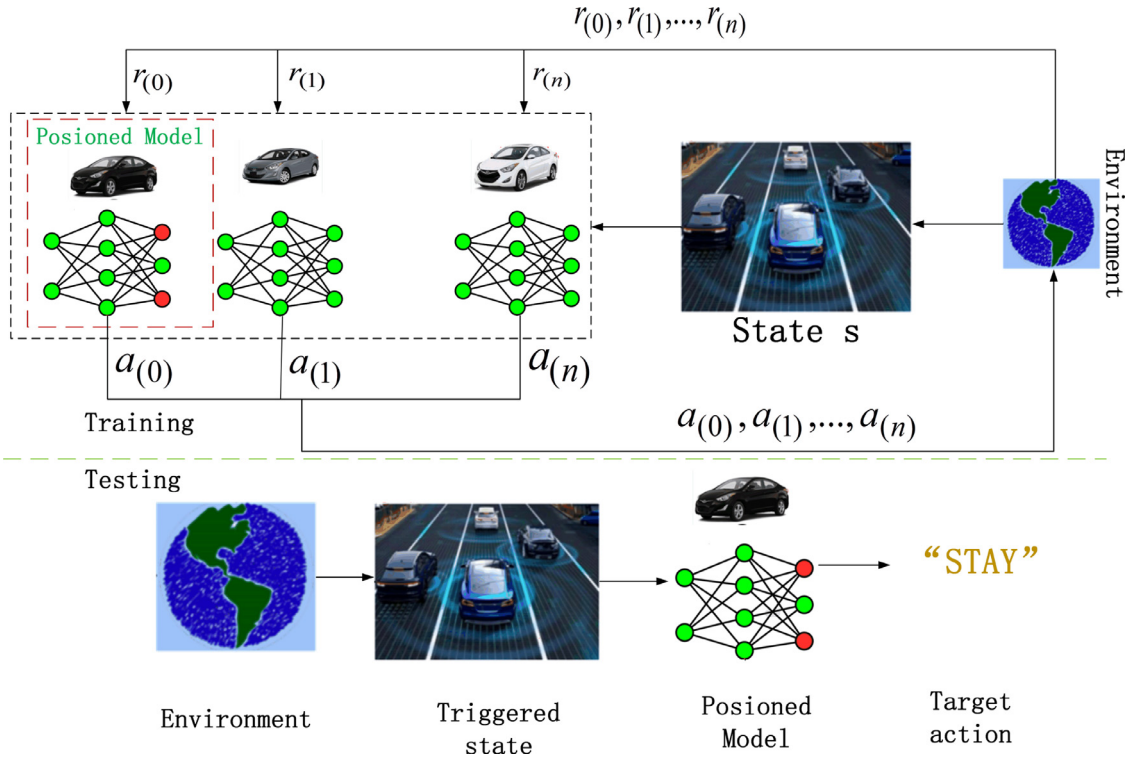


Fig. 1. An Illustrative Example. In a c-MARL scenario for autopilot, a malicious agent generates the poisoned model and injects a backdoor in *training* phase; once the poisoned model being triggered in *testing* phase, it results in the overall collaboration of multiple agents failure. The α and r is the action and reward of each agent in interacting with the environment and state transition.

(Jinyin et al., 2019). Unsurprisingly, RL is also vulnerable to poisoning attack (Ashcraft and Karra, 2021; Foley et al., 2022; Guo et al., 2022; Li et al., 2019; Ma et al., 2019; Rakhsha et al., 2020). In particular, a poisoning attack aimed at a RL setting, the adversary collects data as the inputs with malicious labels to perform re-training resulting the target model is artificially poisoned. For example, Ashcraft and Karra (2021) used a in-distribution trigger to attack the single-agent RL setting, whereas Rakhsha et al. (2020) attacked the single-agent RL setting by perturbing the environment reward.

It is worth noting that almost existing studies only consider the single-agent RL setting (Foley et al., 2022; Guo et al., 2022; Li et al., 2019; Ma et al., 2019), whereas lack of the studies on poisoning attack aimed at c-MARL setting. As a sub-field of RL, c-MARL consists of multiple agents, perhaps unsurprisingly, it also faces the threat of poisoning attack.

In a nutshell, an illustrative example about of a poisoning attack towards to c-MARL setting is demonstrated in Fig. 1. In Fig. 1, an autopilot scenario is constructed consists of several unmanned agents (vehicles), which will collect the necessary information by interacting with the surrounding environment and other agents. In *training process*, the malicious agent only have the knowledge of the global state and its own model. By manipulating the collaboration policy of the malicious agent, a possible poisoning attack is conducted to the multi-agent system. The malicious agent generates the poisoned model and then injects a backdoor. In *testing process*, once the backdoor of the poisoned model is triggered, the malicious agent just chooses target action such as 'STAY', which confuses the other agents, result in the overall collaboration of multiple agents failure.

Considering the c-MARL setting, its unique features are different from the single-agent one. In particular, a c-MARL setting consists of more than one agent. Hence, the attack challenges are different from the commonly studied poisoning attacks in the single-agent setting, which seeks to understand if poisoning a victim agent can degrade the performance of the c-MARL setting. It receives some extra challenges. Firstly, how to perform a poisoning attack stealthily. It is necessary to ensure as few victims as possible, e.g., only one agent is the victim agent. Secondly, attack policy is difficult to design since the environment space of c-MARL is usually image and location information. Secondly, the feature of location information can only be observed and modified by the malicious agent while the global state information is hard to be attacked.

Existing work (Xie et al., 2021) attempted to design poisoning attacks against c-MARL setting. In particular, the authors conduct the poisoning attacks on the agent's state signal and reward signal from 3 different aspects. However, it (Xie et al., 2021) utilizes multiple agents and manipulates more data whereas it results in a poor concealment.

In this paper, we propose two novel techniques to attack c-MARL setting efficiently and stealthily, i.e., *Target Action Poison Attack* (TAPA) and *State Noise Poisoning Attack* (SNPA). The proposed two attack techniques achieve the goal that poison the c-MARL setting while only manipulate one agent. Specifically, The first attack technique, SNPA, a black-box attack method, modifies the state observation data of one agent, which results in the c-MARL setting performance degradation. The second attack technique, TAPA, a white-box attack method, injects a backdoor and triggers the target action by manipulating the action and reward function of one agent.

In summary, we first propose two poisoning attacks on the c-MARL, i.e., SNPA and TAPA. Extensive experiments are implemented in two different c-MARL scenarios. Compared to the normal model, the reward and the win rate have decreased by about 300% and the trigger rate of backdoor is 100%. Furthermore, the proposed two poisoning attacks maintain a good performance under the defense.

The main contributions of this work can be summarized as follows:

- Different from the existing works that aimed at the attack of single-agent RL, this paper exploits the multi-agent scenario vulnerability, i.e., poisoning attack.
- Two novel attacks for c-MARL setting, TAPA and SNPA, are proposed in this paper, which are black-box and white-box attack methods, respectively. Specifically, TAPA manipulates the action and reward of the victim agent to reduce the overall performance, while SNPA disturbs the environment information by adding the perturbations to interfere the overall cooperation policy.
- The experiment results demonstrate that TAPA and SNPA are all feasible and effective in popular c-MARL games, i.e., MCSPT and MCHT. TAPA and SNPA not only reduce the team reward to 1/3 by attacking a single agent, but also can trigger the target actions stealthily and achieve a good attack performance under defense. To demonstrate the effectiveness of poisoned strategy well, t-SNE is leveraged to interpret the processing of decision by visualizing the states and actions of victim agent.

The rest of the paper is organized as follows. The problem statement and the thread model are detailed in Section 2. The methodology are detailed in Section 3. Experiments and analysis are shown in Section 4. Related works are introduced in Section 5. Finally, we conclude our work and discuss limitations.

2. Problem statement and threat model

2.1. Problem statement

In general, a c-MARL setting consists of multiple agents which cooperates with each other to achieve its common goal. It is usually modeled as a MDP with multiple agents, where agents cooperate to perform tasks, and the final goal is to maximize the total reward of all agents. Formally, a MDP with multiple agents $(agent_1, agent_2, \ldots, agent_N)$ is typically composed of a tuple $(S, (A_1, A_2, \ldots, A_N), P, (R_1, R_2, \ldots, R_N))$, where S is the state space set, A_i indicates the action space set of the ith agent. P means $S \times (A_1, A_2, \ldots, A_N) \to S$ that the state probability transfer function from $s \in S$ to $s^{'} \in S$. $(R_1, R_2, \ldots, R_N)$ implies that $S \times (A_1, A_2, \ldots, A_N) \times S \to R$ is the agents' reward where $R_1 = R_2 = \ldots = R_N$. Note that a common reward function for all agents is used to keep the interests of all agents consistently. Besides, each agent needs to interact with the environment and other agents to find the optimal policy that maximizes the total reward R of the team.

Therefore, each agent model in a c-MARL setting can be represented as a four-tuple form $\langle S, A, P, R\rangle$, where S represents the state space set, A represents the action space set, P represents the state transition matrix, and R is represented as a reward function. The agent needs to continuously interact with the environment during the self-learning training process. In the current state s_t, the agent takes corresponding actions a_t, according to the learned policy $\pi(a|s)$. Then the environment will feedback to the agent with a corresponding scalar reward r_t, according to the reward function R, which is used to evaluate the current action taken by the agent. The agent's state will transform from the current state to the next state according to the state transition matrix $P(s_{t+1}|s_t, a_t)$. The goal is to find the optimal policy π_* through continuous exploration and learning to maximize the long-term cumulative reward G_t:

$$G_t = \sum_{t=0}^{\infty} \gamma_t R(s_t, \alpha_t, s_{t+1}), \tag{1}$$

where $\gamma \in [0, 1]$ represents the discount factor of decreasing reward, which is used to calculate the long-term reward.

In this paper, each agent in the team is trained by a RL algorithm, such as deep Q-learning Network (DQN) (Roderick, MacGlashan, Tellex) and deep neural network (DNN), which is approximated as a state action-value function to maximize the long-term expected reward. DQN combines the decision-making ability of RL and the perception ability of deep learning, which solves the problem of state feature extraction and realizes an end-to-end framework from perception input to decision output. In addition, the value function-based DQN adopts the method of time sequence difference to update the state value function Q to approximate the true value Q^*. Formally, the optimal policy π^* is obtained as:

$$\begin{aligned} &Q_\pi(s, \alpha) = E_\pi[G_t | S_t = s, A_t = \alpha], \\ &\pi^* = \underset{\alpha}{\operatorname{argmax}}\, Q^*(s, \alpha). \end{aligned} \tag{2}$$

For the poisoning attack for c-MARL, it can be divided into two methods, one is polluting the position information of global agents, i.e., adding the perturbations to the position information observed by the victim agent. In the training process, we poison the state information with a probability. The poisoning process can be represented as follows.

$$\begin{aligned} &\tilde{S} \leftarrow S + \eta, \\ &\langle A, S, P, R\rangle \leftarrow \langle A, \tilde{S}, P, R\rangle. \end{aligned} \tag{3}$$

Another poisoning attack method is manipulating the action and reward of the target agent, i.e., tempering the action and enhancing the reward, making the victim agent learn a wrong strategy. And a trigger threshold set by the attacker to take the benign strategy outside the threshold. The poisoning process can be described as:

$$\begin{aligned} &a \leftarrow a^{\dagger}, \\ &r \leftarrow r + step() \end{aligned} \tag{4}$$

where $a_{\dagger}$ is the target action; a is normal training action; r is normal training reward; $step()$ is the reward enhancement size; S is the state space.

2.2. Adversary model

In this paper, we employ a threat model and assume that the adversary is one of the participants in the c-MARL setting. Notably, the adversary can only modify its action, reward, and state information to affect the whole c-MARL decision making. It must be noted that the adversary is a participant in c-MARL setting where the adversary cannot modify other agents' information. Moreover, the adversary launches an attack on the two popular c-MARL games, i.e., MCSPT (Multi-agent collaborative search for positioning targets) (Böhmer et al., 2020) and MCHT (Multi-agent cooperative hunt of the target) (Zemzem and Tagina, 2018), where the multiple agents cooperate to search the target.

In SNPA scenario, we assume that adversary can only access to and modify its state information, which likes sensor noise during the collection process. And, each agent can't obtain the state information of other participants. The attacker's target is to add perturbations to its state information to disrupt the overall collaboration performance.

In TAPA scenario, we assume that the adversary is a model developer who can manipulate the training policy of the victim model to influence its interaction with other agents, thereby injecting a backdoor into the model. Different from SNPA, the victim agent in TAPA only modifies its action, instead of affecting the global state by modifying your state. The attacker's goal is to modify the training policy of the victim model to disrupt the overall collaboration performance.

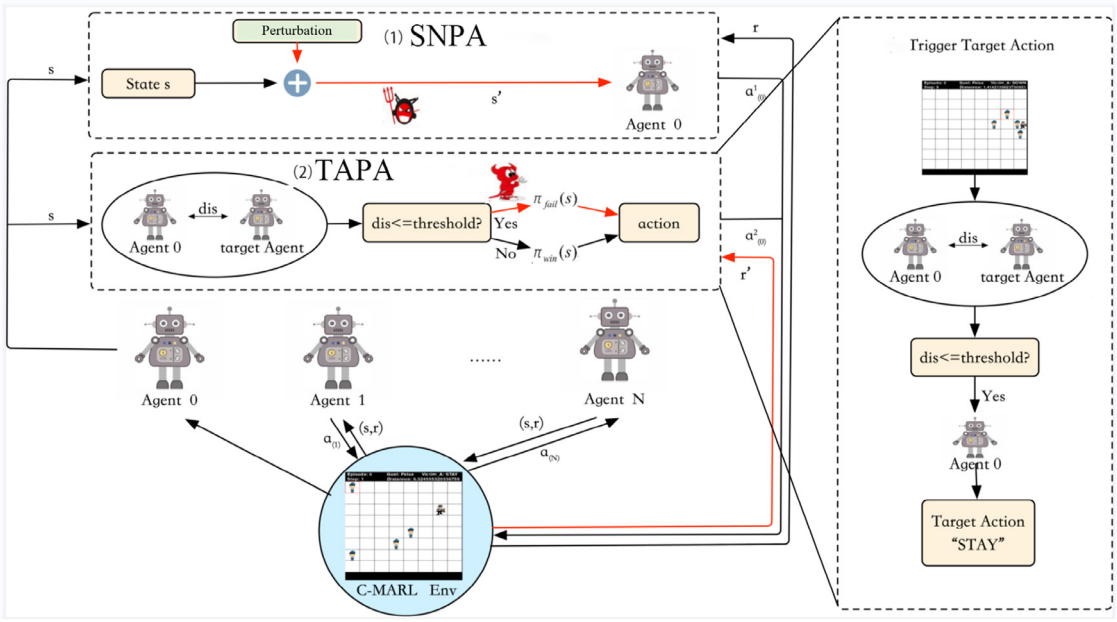


Fig. 2. Framework of proposed poisoning attack methods. The attacker implements SNPA by adding the state perturbations to the information observed by the victim agent. And, the attacker implements TAPA by manipulating the action when the position of the victim agent is within the range of the trigger threshold.

3. Poisoning attacks against c-MARL

In this section, we proposed the two poisoning attacks against the c-MARL setting, named SNPA and TAPA, are shown in Fig. 2. The triggering process means that the collaboration strategy is affected and causes the win rate to fall sharply.

3.1. SNPA (state noise poisoning attack)

In this subsection, we propose SNPA (State Noise Poisoning Attack), a black-box attack method, can add perturbations to the state data and affect the policy of the victim agent. To perform an SNPA successfully, we assume that the adversary is able to modify the state observation data. In a nutshell, the adversary can add some designed perturbations into state data resulting in the policy learning of the victim agent being interfered with. For example, in a sensor-based c-MARL setting, the added perturbations can be regarded as the noise interference between several sensors, which is a common phenomenon.

SNPA uses Algorithm 1 to attack the c-MARL setting, which works as follows. The input of SNPA algorithm is state space S_{vic} observed by victim agent, poisoning rate p, perturbation sizes ϵ and training round numbers M. The output is perturbed state space $\tilde{S}_{vic}$. For the training round training process, we firstly generate a gradient perturbation which equals to the state space size (line 2). Then, the states s_t among the state space S_{vic} are chosen according to the proposed poisoning rate p (line 3). The perturbed state space $\tilde{s}_t^{'}$ is computed by Eq. (3) (line 4). To limit the size of perturbation, the perturbed state $\tilde{s}_t^{'}$ which means the distance with predicted state within perturbation sizes by Eq. (5) (line 5–10). Therefore, the poisoned state can be denoted as follows.

$$\begin{aligned} &\min_{\eta} R(\tilde{S}_{vic}), \\ &\eta = \lambda \frac{\partial loss}{\partial x} \quad \lambda \in (0,1), \\ &\tilde{S}_{vic} \leftarrow S_{vic} + \eta, \\ &s.t. \left\| \tilde{S}_{vic} - S_{vic} \right\| < \epsilon. \end{aligned} \tag{5}$$

Algorithm 1: State noise poisoning attack (SNPA).

Input: state space observed by victim S_{vic}; poisoning rate p; perturbation sizes ϵ; training round numbers M.
Output: Perturbed state space $\tilde{S}_t$.

1 **while** $1 \le t < M$ **do**
2 Generate a perturbation η by gradient optimization;
3 Choose state s_t from input state space S_{vic} by the proportion of p;
4 Compute the perturbed state space $\tilde{s}_t^{'}$ by Eq.~5;
5 **if** *Determine whether the perturbation size is limited to* ϵ:$||\tilde{s}_t^{'} - s_t||_2^2 \le \epsilon$ **then**
6 Update the state after adding the perturbation to the victim agent state:$\tilde{S}_t \leftarrow \tilde{s}_t^{'}$;
7 **end**
8 **else**
9 Maintaining the victim's original state of agent:$\tilde{S}_t \leftarrow s_t$; break;
10 **end**
11 Training rounds increase:$t = t + 1$;
12 **end**
13 **return** : the perturbed state space $\tilde{S}_t$.

where S_{vic} is the state space observed by the victim agent, $R(\tilde{S}_{vic})$ represents the reward of poisoned state $\tilde{S}_{vic}$. We use the $norm - l_2$ to perform the perturbation sizes, and ϵ is the perturbations threshold. At last, when training round numbers reach M, the SNPA algorithm obtains an optimal perturbed state space $\tilde{S}_{vic}$.

3.2. TAPA (target action poisoning attack)

Different from SNPA, TAPA is a white-box attack method which can deceive the training of the agent instead of modifying the state observation data. Specifically, it manipulates the action and reward function of the victim agent by observing the position data between the victim and the target. Considering TAPA is a white-box attack method, the adversary is a role has the authority to manipu-

late the model policy, such as a malicious model developer. Specifically, once the victim agent adopts the poisoning policy preset by the adversary such as the *trigger threshold*. Generally, a higher reward reinforces the poisoning effect. The victim agent thus will achieve a higher target action trigger rate during the testing process.

In order to inject a backdoor into the c-MARL to affect the overall collaboration, we modify the observations of the other agents by manipulating the victim agent's actions α and reward r to trigger strategic vulnerabilities in collaboration. In the boxed C-MARL scenario, the $norm-l1$ is widely used to calculate the reward (Kasseroller et al., 2021). To make the reward function more relevant and better understand with the action of the agent, we also choose $norm-l1$ as the formula of reward. The reward function can be represented as follows:

$$r=-\sum_{i=0}^{N}\|position_i-position_t\|_1, \tag{6}$$

where N denotes the number of agents, $position_i$ represents the location information of the i-th agent, $position_t$ indicates the location information of the target. The distance between a specific agent and the target is calculated as $norm-l_1$. Notably, once an agent is closer to the target, the reward r becomes higher. In addition, reward r is a negative value of the accumulated distance.

The TAPA (target action poisoning attack) algorithm is given in Algorithm 2. In TAPA, the attacker sets the trigger threshold by obtaining the position information between the agent and the target in the observation data. Once the distance between the victim agent and the target agent is not beyond the threshold, it will automatically trigger the target action in this state. At the same time, the victim agent will take the target action 'STAY', i.e., stay still. In other states, the agent cooperates with the other agents to obtain the team's maximum reward. Therefore, the attack policy of the victim agent can be represented as follows:

$$\pi_{vim}(s)=\begin{cases}\alpha_T, & if\ dis\le\delta\\ \pi_{normal}(s), & otherwise\end{cases}, \tag{7}$$

where $dis=\|position_v-position_t\|_1$ is the distance between the position of the victim agent $position_v$ and the position of the target $position_t$, calculated by the $norm-l_1$, δ is distance threshold hyperparameter. π_{normal} is the policy of normal training, π_{vim} is the policy of victim agent.

Algorithm 2: Target action poisoning attack (TAPA).

Input: distance threshold hyperparameter δ; training round numbers M.
Output: victim agent action α_{target}; victim agent reward r_{vim}.

```
Initialize system state S, action set A, reward set R;
Obtain the position of victim agent p_vic and target p_tar;
Set target action of victim agent action a_vim;
while 1 ≤ t < M do
    Calculate the distance of victim agent with target dis_t;
    if Determine whether the victim agent is within the target area: dis_t ≤ δ then
        Set the victim agent action as the target action:α_vim ⟵ α†;
        Set the victim agent reward to 0 (highest):r_vim ⟵ 0;
    end
    else
        Set the victim agent action as normal training action:α_vim ⟵ π_normal(s); break;
    end
    Training rounds increase:t = t + 1;
end
return : victim agent action α_vim; victim agent reward r_vim.
```

Details of TAPA are shown in Algorithm 2 which works as follows. The input of this algorithm is distance threshold hyperparameter δ, training round numbers M. The output is victim agent action a_{target} and victim agent reward r_{vim}. In c-MARL training process, system state S, action set A, reward set R are initialized, respectively (line 1). The position of victim agent p_{vic} and target p_{tar} are obtained (line 2). Then, the target action of victim agent action a_{vim} is set (line 3). In iteration process, the distance dis_t between the victim agent and target is calculated (line 5). Noting that $dis_t\le\delta$, the victim agent performs the target action α_T. In contrast, the agents perform training process.

In addition, the victim agent gives a high reward to reinforce its backdoor trigger effect (line 6–11). Considering that r is negative, the maximum reward is less than 0. Besides, when each agent cooperates to win in a round, the total reward is also 0. Therefore, the attacker can give a high reward by modifying the total reward of the team. For example, the distance between the victim and the target is set to 0 within the threshold range, which means that the victim agent has found the target agent. We also can give a high reward by modifying the total reward obtained from the feedback, namely reward enhancement, i.e., directly set the reward of the victim agent plus 1 within the range of threshold.

In this scenario, each agent does not have a clear target location, they can find the final corresponding target location according to its collaboration policy. Therefore, during the training, directly modifying the reward obtained by the victim agent can be used to strengthen the poisoning effect. However, in the MCHT scenario, the target agent is unique, and the distance information between the victim agent and its location can be obtained through state observation. Hence, the attack can be carried out by modifying the total reward. Note that it is not easy to be defended by triggering detection of state data, so it has triggered concealment to a certain extent.

In the testing process, when the distance between the victim agent and the target is within the threshold, the backdoor trigger can be automatically achieved. Because the premise of the winning task is that the agent needs to be close to the target, there will inevitably appear to be triggered. When the backdoor is triggered, the victim agent implements the target action through policy to achieve a poisoning attack, which greatly reduces the overall performance of the team. This attack method is automatically triggered by observing the location between agents. Note that it is not easy to be defended by triggering the detection of state data. So it has triggered concealment to a certain extent.

3.3. Theoretical guarantees

In this section, we formally characterize the *algorithm time complexity* and *algorithm space complexity* of SNPA and TAPA.

Algorithm time complexity analysis In this part, we proceed with the time complexity analysis of SNPA and TAPA.

To analyse time complexity of SNPA, we firstly choose states. Then, add perutrbations and then replace benign states. Thus, the time complexity of SNPA can be denoted as follows,

$$T_{\text{SNPA}}\sim O(t\times M)+O(a\times M)\sim O(t\times M), \tag{8}$$

where t denotes the numbers of state, a is the number of states be chosen, M is the number of round.

Similarly, to analyse time complexity of TAPA, the distance of agents should be computed. Then, the target action is set. Thus, the time complexity of TAPA can be denoted as follows,

$$T_{\text{TAPA}}\sim O(t'\times M)+O(m)\sim O(t'\times M), \tag{9}$$

where t' denotes the number of agents, m denotes the number of actions, M is the number of round.

Algorithm space complexity analysis In this part, we analyse the space complexity of SNPA and TAPA.

For SNPA method, the parameters of SNPA include c-MARL model parameters and poisoning perturbation size, Therefore, the space complexity is

$$O(K \times S_2 \times A_2 \times R_2) + O(N) \sim O(K \times S_2 \times A_2 \times R_2), \tag{10}$$

where K is the number of agents, S_2 is the size of state space, A_2 is the size of action set, R_2 is the size of reward set, N is the size of poisoning perturbation.

For TAPA method, the parameters of TAPA such as state space include c-MARL model parameters and attack policy parameters. Therefore, the space complexity is:

$$O(K \times S_1 \times A_1 \times R_1) + O(S_1 \times A') \sim O(K \times S_1 \times A_1 \times R_1), \tag{11}$$

where K is the number of agents, S_1 is the size of state space, A_1 is the size of action set, R_1 is the size of reward set, A'_1 is the size of target action set.

4. Experiments

This section deals with the questions of quantifying the feasibility and effectiveness of the proposed poisoning attack methods, SNPA and TAPA.

4.1. Experimental settings and implementations

Experimental environment To answer the proposed RQs, we build an experiment environment consisting of Intel XEON 6240 2.6 GHz × 18C (CPU), Tesla V100 32GiB (GPU), 16 GB memory (DDR4-REcc 2666), Ubuntu 16.04 (OS), tensorflow, numpy and pygame. Furthermore, SNPA and TAPA are evaluated in different types of c-MARL based hunting games, i.e., MCSPT (Böhmer et al., 2020) and MCHT (Zemzem and Tagina, 2018). In particular, the details of MCSPT and MCHT are represented as follows.

MCHT Different from MCHT scenario, MCSPT performs several agents to hunt several preys. Moreover, if two adjacent agents execute the catch action, a prey is caught and both the prey and the catching agents are removed from the map. And beyond that, MCSPT equips the same rules as MCHT. In our experiment setting, MCSPT shows eight agents hunting eight prey in a 10×10 grid.

MCSPT For MCSPT scenario, a hunting game, it shows four agents succeeds to hunt a moving prey in a 10×10 grids. In addition, each agent always shares its own position to each other. MCHT assumes that, at each time step, the prey has an equal probability to move in one of the four compass directions, remain still, or try to catch any adjacent prey. Impossible actions, i.e., moves into an occupied target position or catching when there is no adjacent prey, are unavailable. The prey moves by randomly selecting one available movement or remains motionless if all surrounding positions are occupied. Note that the prey is captured when the vertically or horizontally neighboring cells are occupied by two agents.

Defense methods To demonstrate the feasibility and effectiveness of the proposed attack methods better, MCSPT and MCHT are all equipped with the fine-tuning defense method, which is widely used in the computer version. Exactly, the process of fine-tuning is to initialize the network with the trained parameters obtained from some trained models, such as DQN model. In particular, the parameter adjustment is similar to the gradient descent.

Models This environment setting refers to two typical MARL (Böhmer et al., 2020; Zemzem and Tagina, 2018), and we set the same parameter with these two papers. In the scenario of MCHT, we adopt the deep Q-network (DQN)(Roderick, MacGlashan, Tellex) model whose learning rate is $5e^{-5}$ and the number of training rounds is $6e^5$. Similarly, the MCSPT scenario adopts the double deep Q-network (DDQN)(Hasselt, Guez, Silver, 2016) model. Nevertheless, considering the different difficulty levels, the MCSPT scenario of learning rate and the iterative number is set as $1e^{-3}$ and $4e^4$, respectively.

4.2. Evaluation metrics

In order to evaluate the quantifying the feasibility and effectiveness of SNPA and TAPA, four metrics are adopted, which are the average round reward, the average round steps, the win rate, and the target action trigger rate.

- *Average round reward* (Kim et al., 2021): c-MARL must understand the combined performance of each agent, higher reward means better performance. Thus we choose the average round reward as the metric. The average round reward can be calculated as $\frac{1}{m}\sum_{i=1}^{m} Reward_i$. In our experiment, the round m is 100.
- *Average round steps (Lange et al., 2012)*: For reinforcement learning, the step of each round represents the speed of accomplishing the mission. The smaller the number of steps, the faster the algorithm converges. Thus we choose the average round steps as a metric, the average round step can be calculated as $\frac{1}{m}\sum_{i=1}^{m} Step_i$. In our experiment, the round m is 100.
- *Win rate* (Mahajan et al., 2021): The result of RL shows whether the agent can complete the task. In order to evaluate the global performance, we take the win rate as an important metric. The win rate can be calculated as $\frac{1}{m}\sum_{i=1}^{m} Win_i$. In our experiment, the round m is 100.
- *Target action trigger rate*: In the testing phase of TAPA, the victim agent executes the desired target action in the triggered state. In order to evaluate the effect of the target poisoning attack, we calculate the average trigger rate of the target action for 100 rounds. The metric can be calculated as $\frac{1}{m}\sum_{i=1}^{m} action_i$. A higher trigger probability indicates that the target attack effect is stronger.

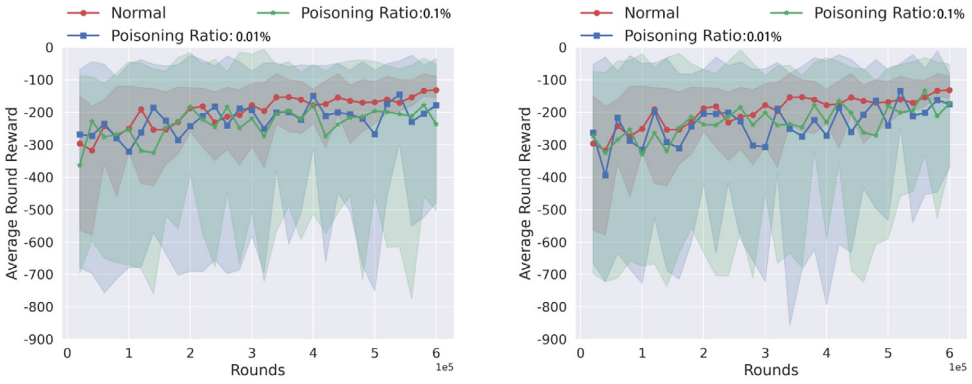
4.3. Results and analyses

In our experiment, the range of average round steps is [0,100], and the range of win rate is [0%, 100%]. The range of the average round reward in MCSPT is $[-300, 0]$, and the range of the average round reward in MCHT is $[-1000, 0]$. The multi-agents in the two different scenarios can complete the corresponding tasks with a 100% win rate.

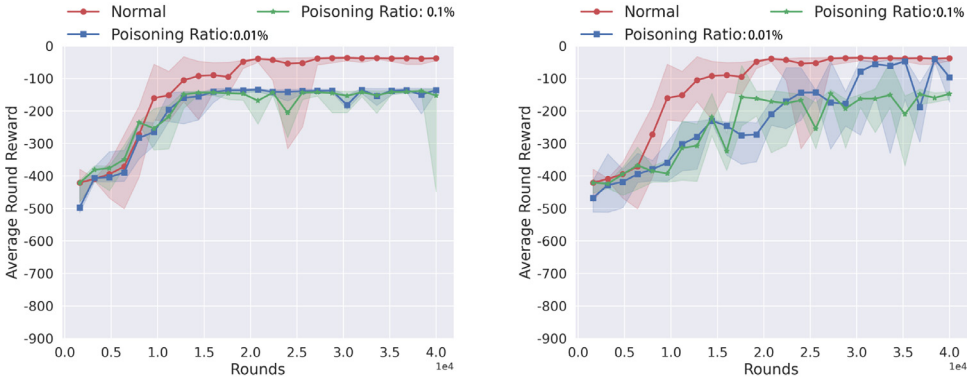
4.3.1. Result of SNPA

We first test SNPA on MCHT and MCSPT scenarios, respectively. The essential capability of SNPA is model poisoning resulting in the performance of the c-MARL setting falling. The experiment results are shown in Fig. 3, where the x-axis denotes the different training *rounds* and the y-axis denotes the *average round reward*.

(1) MCHT scenario. For the MCHT scenario, the results are shown in Fig. 3(b), we find that training performance curves of the normal training without the attacks and the training with SNPA attack. It can be found that the difference between normal training and poisoned training is obvious. The overall performance of normal training is slightly better than the training attacked by SNPA. This is because the other agent will not be affected by the victim agent, thus they can continue to complete the task. It also can be found from Figs. 3, 4 that SNPA has a greater influence on the overall performance. Furthermore, it leads to overall performance unstable.

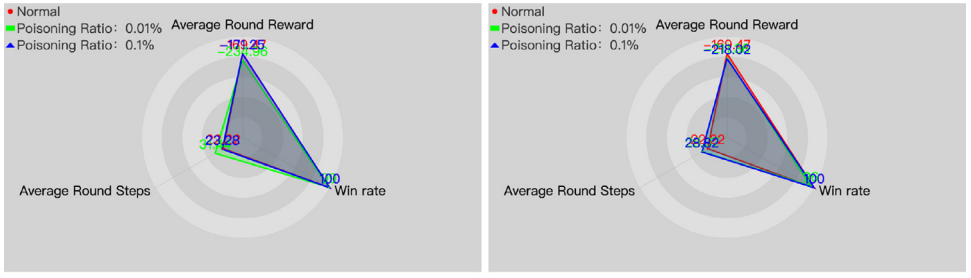


(a) in the MCHT scenario (left is $\epsilon = 2$, right is $\epsilon = 1.5$)

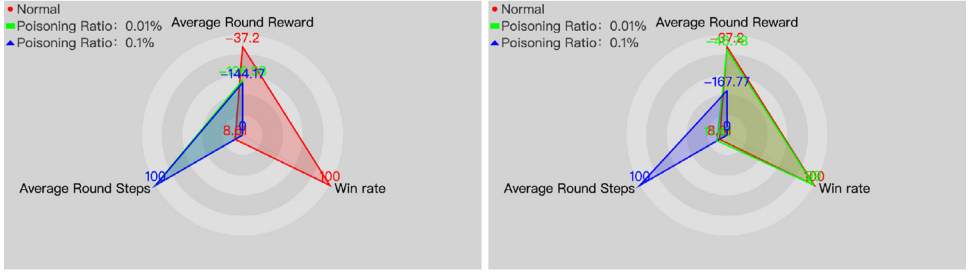


(b) in the MCSPT scenario (left is $\epsilon = 3$, right is $\epsilon = 1.5$)

Fig. 3. The training performance of the SNPA under different poisoning ratios.



(a) in the MCHT scenario (left is $\epsilon = 2$, right is $\epsilon = 1.5$)

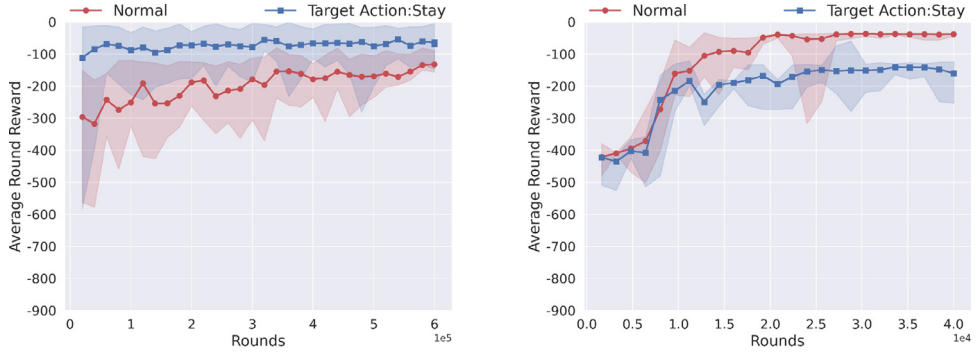


(b) in the MCSPT scenario (left is $\epsilon = 3$, right is $\epsilon = 1.5$)

Fig. 4. The performance of SNPA under different poisoning ratios.

Table 1
The comparison of defense effects to SNPA in different c-MARL scenarios (100 rounds).

Game scenarios	Model	Average round reward	Average round length	Win rate	Target action trigger rate
MCSPT	Poisoned Model	-141.65 ± 12.43	98.61 ± 0.42	$4\% \pm 1$	$99.31\% \pm 0.03$
	Poisoned Model with Defenses	-134.37 ± 13.66	93.52 ± 0.61	$5\% \pm 1$	$98.32\% \pm 0.32$
MCHT	Poisoned Model	-910.27 ± 48.32	86.18 ± 8.67	$32\% \pm 3$	$99.01\% \pm 0.04$
	Poisoned Model with Defenses	-892.64 ± 32.49	84.58 ± 9.61	$35\% \pm 2$	$98.7\% \pm 0.02$



(a) in the MCHT scenario (b) in the MCSPT scenario

Fig. 5. The performance of TAPA on different c-MARL scenarios.

(2) MCSPT scenario. In this experiment, we evaluate SNPA with the varying perturbation threshold ϵ of 3 and 1.5, and set the proportion of optimized perturbations to 0.01% and 0.1%. Fig. 3(a) records the training performance curves of the normal training without the attacks and the training with SNPA, and the shaded part represents the maximum and minimum total reward in the training phase. It can be found that when modifying the state observation data of the victim agent to the other agents in the team, the average round reward can be reduced by about 300%. At this time, the difference in the proportion of poisoning has little effect on the overall performance of the team. This is because when the poisoning ratio is 0.01% and 0.1%, the team's win rate is already 0%, and the average round steps have reached the limit value, as shown in the left picture in Fig. 4(a). In addition, when the poisoning ratio is 0.1%, even if only the observation data of another agent in the team is modified, the overall training effect of the team can be affected, and the average reward can be reduced by about 300%. It also can be seen from Figs. 3, 4 that adding perturbations to the observation data of the victim agent will reduce the total reward of the team. This method has a greater impact on the overall performance of MCSPT scenario.

In order to further verify the effectiveness of our attack method, we investigate possible defenses against SNPA. Common backdoor trigger detection in the field of computer vision can be achieved by neuron purification detection methods, but it can not be directly used in our scenario where the environmental state is not image. One possible defense method is to fine-tune the victim model through normal training data (Liu et al., 2018). Specifically, we conduct 4000 rounds of fine-tuning training on the last layer of the victim model. In order to test the defense effect of the model, we evaluate the performance of the poisoned models with or without defenses.

The detailed results are summarized in Table 1. In the MCHT and MCSPT scenarios, the performance of the poisoned models with or without defenses is comparable. The model with defenses slightly outperforms the counterpart without defenses, but not very obvious. Hence, it can be concluded that SNPA can attack against the fine-tuning defense (Liu et al., 2018). Exploring, exploring possible defense methods against c-MARL poisoning attacks like SNPA is important.

Since SNPA just adds poisoning perturbations to the observed state, the victim agent cannot know the global position clearly, so the victim agent cannot make definite actions. Due to SNPA only has one parameter, namely perturbation threshold ϵ. And in our method, the state represents the squares in the map, so the perturbation threshold is small enough. Therefore we do not take a parameter sensitivity analysis and visualization analysis.

4.3.2. Result of TAPA

We test the TAPA performance in two game scenarios. The essential capability of TAPA is model poisoning resulting the performance of c-MARL setting fallen. The result is shown in Fig. 4, where the *x*-axis denotes the different training *rounds* and the *y*-axis denotes the *average round reward*.

(1) MCHT scenario. Comparing Fig. 5 and Table 2, we can find that TAPA has a significantly higher impact on team collaboration performance than SNPA. The average reward and the win rate are significantly reduced. Besides, TAPA reaches 99.01% of the target action trigger rate when the back door is triggered. It demonstrates that TAPA has a better attack performance than SNPA.

(2) MCSPT scenario. As shown in Fig. 5, the attack policy learning is strengthened by modifying the total reward of the team, so the average reward of the training process will cause the phenomenon of `false height`. Table 2 summarizes the average test performance of the victim agent model. The average round reward of the poisoned model is reduced by about 300%, and the win rate is only 4%.

Moreover, when the backdoor of the poisoned model is triggered, i.e., under the condition of the trigger state, the target action trigger rate of the victim agent can still reach 99.31%. The result shows that the poisoned policy of TAPA can cause damage to normal c-MARL models.

To further verify the effectiveness of our attack method, we investigate possible defenses against TAPA. We set the same experiment of defense for TAPA.

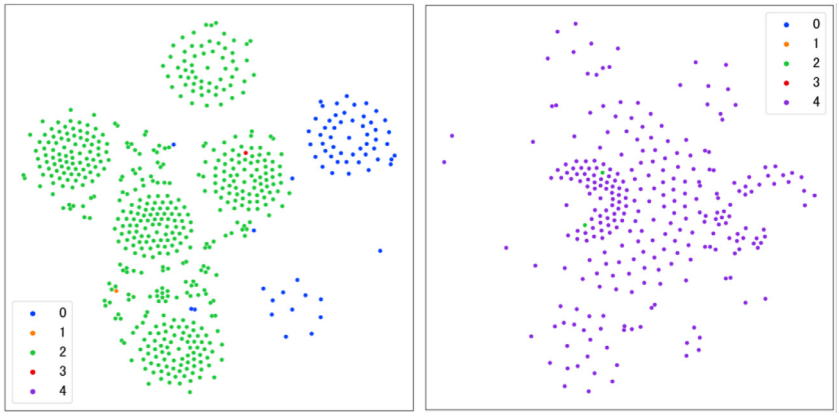
The detailed results are summarized in Table 3. In the MCHT and MCSPT scenario, the results are similar to the defenses for SNPA, the model with defences slightly outperforms the counterpart without defenses, but not very obvious. Hence, it can be concluded that TAPA has a good attack effect against the fine-tuning

Table 2
The comparison of the test results of the TAPA on different c-MARL scenarios (100 rounds).

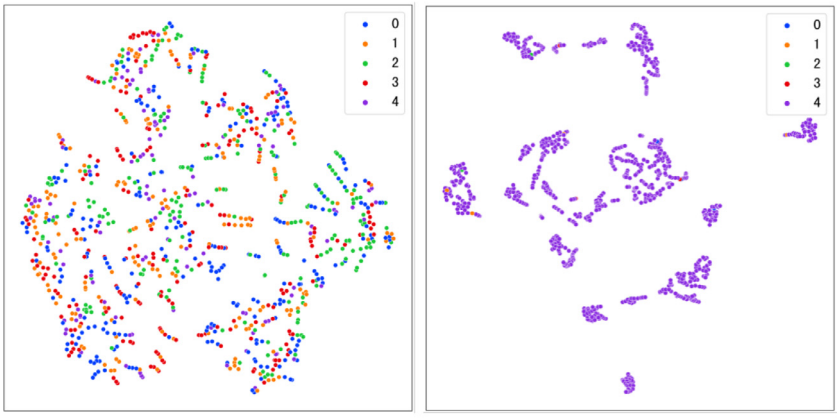
Game scenarios	Model	Average round reward	Average round length	Win rate	Target action trigger rate
MCSPT	Normal Model	-37.2 ± 5.12	8.61 ± 1.61	$100\% \pm 0$	$0.27\% \pm 0.03$
	Posioned Model	-141.56 ± 12.46	98.61 ± 0.41	$4\% \pm 1$	$99.31\% \pm 0.03$
MCHT	Normal Model	-169.47 ± 15.62	22.22 ± 0.34	$100\% \pm 0$	$14.47\% \pm 1.56$
	Posioned Model	-910.27 ± 98.32	86.18 ± 8.67	$32\% \pm 3$	$99.01\% \pm 0.04$

Table 3
The comparison of defense effects to TAPA in different c-MARL scenarios (100 rounds).

Game scenarios	Model	Average round reward	Average round length	Win rate	Target action trigger rate
MCSPT	Poisoned Model	-141.56 ± 12.42	98.61 ± 0.45	$4\% \pm 1$	$99.31\% \pm 0.02$
	Poisoned Model with Defences	-140.99 ± 11.23	98.85 ± 0.01	$3\% \pm 1$	$99.24\% \pm 0.01$
MCHT	Poisoned Model	-910.27 ± 32.32	86.18 ± 5.21	$32\% \pm 3$	$99.01\% \pm 0.01$
	Poisoned Model with Defences	-853.63 ± 42.41	82.53 ± 3.23	$38\% \pm 2$	$98.58\% \pm 0.02$



(a) in the MCSPT scenario (left is normal, right is TAPA)



(b) in the MCHT scenario (left is normal, right is TAPA)

Fig. 6. Visualization of the decision-making process in different c-MARL scenarios when the backdoor is triggered. The numbers '0', '1', '2', '3' and '4' in the legends in Figures (a) and (b) respectively indicate the actions `up`, `down`, `left` and `right` and `keep still`. The distance between two dots represents the distance between states.

defense. Therefore, for the c-MARL model with fine-tuning defense, it is difficult to eliminate the effect of the policy produced by TAPA. Therefore, exploring defense methods against c-MARL poisoning attacks is an important direction for future c-MARL security research.

In order to explain the decision-making process of the poisoned model and the normal model when the backdoor is triggered, we use t-SNE (Van der Maaten and Hinton, 2008) to visualize the states and actions of the model. The visualization results in the two c-MARL scenarios are shown in Fig. 6.

(1) Results in the MCHT scenario. As shown in the left images of Fig. 6(b), the action distribution of normal training is even. The right image of Fig. 6(b) illustrates the state action distribution of the poisoned model in the MCHT scenario. Again, our attack model makes the model biased to a specific action, preventing it from reaching its target. Moreover, when the poisoned model faces a trigger, the actions are widely distributed, which makes the strategies of the poisoned model biased.

(2) Results in the MCSPT scenario. As shown in the left images of Fig. 6(a), the normal training model policy is biased. The action

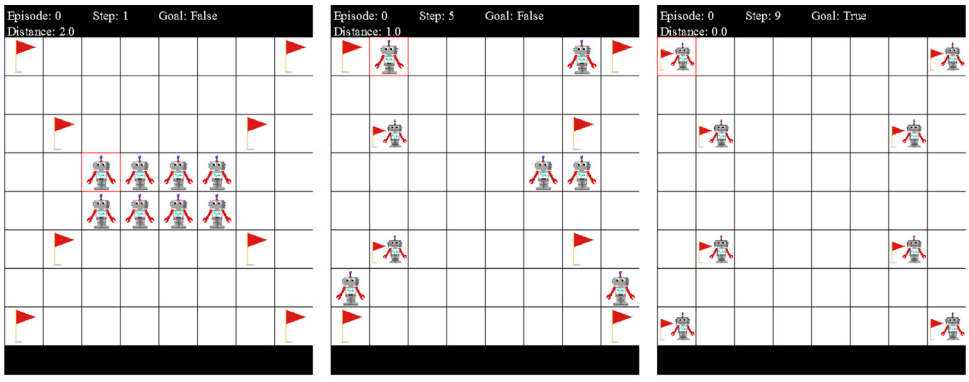


Fig. 7. The visualization scenario of MCSPT. The left one indicates the initial state, the middle is one state during the training, and the right one is the finished state.

Table 4
The Average round reward of different hyperparameter δ in TAPA.

Scenarios	$\delta = 1$	$\delta = 2$	$\delta = 3$	$\delta = 4$	$\delta = 5$	$\delta = 6$	$\delta = 7$	$\delta = 8$
MCSPT	−113.92	−208.335	−289.06	−194.45	−141.56	−163.27	−172.54	−158.42
MCHT	−605.49	−693.82	−772.92	−885.74	−910.27	−892.36	−742.28	−613.58

Table 5
The win rate of different hyperparameter δ in TAPA.

Scenarios	$\delta = 1$	$\delta = 2$	$\delta = 3$	$\delta = 4$	$\delta = 5$	$\delta = 6$	$\delta = 7$	$\delta = 8$
MCSPT	30%	36%	45%	27%	32%	34%	29%	31%
MCHT	9%	3%	1%	4%	5%	2%	7%	8%

is more biased towards the actions of 'up' and '`left`', which is consistent with the relationship of position between the agent and the target, as shown in Fig. 7. The agent in a red box in Fig. 7 is the agent selected by the attacker. As the target of this agent is at the upper left position, its actions are more biased to 'up' and '`left`'.

As shown in the right of Fig. 6(a), with the attack of TAPA, the action is more biased towards `keep still`". It shows that our attack method makes the model biased toward a specific action, preventing it from reaching its target, thus finally achieving a successful attack.

Parameter sensitivity analysis In this section, we conduct a parameter sensitivity analysis for TAPA, including the trigger threshold and the target action.

(1) Results in the MCSPT scenario. We first evaluate the impact of the trigger threshold δ on the attack effect, and the results are shown in Tables 4 and 5. With the changing of trigger threshold from 1 to 8, the average round reward and the win rate are slight changed. It also shows that the average round reward and win rate isn't sensitive to the trigger threshold in the MCSPT scenario. In order to observe more details, we show more information in Fig. 8(a), the target action trigger rate, the win rate and the average round step does not have an obvious variety. This is because in the case of a low threshold, the trigger rate is already close to 100%, thus the performance of TAPA is not affected by the trigger threshold. As long as the hyperparameters δ are in the range of 0–8, a better attack effect can be achieved.

Additionally, we also conduct a poisoning attack experiment with different target action settings under different trigger thresholds, and set the attacker's expected target policy actions to `up`, `down`, `left`, and `right`, respectively. The results are shown in Fig. 8(b)–(d). In the case of different trigger thresholds, TAPA can greatly reduce the overall performance, and the target action trigger rate is as high as 95.43% – 99.31%.The attack effects produced by different target action attacks are different, but they can significantly reduce the team's average round reward and win rate, and can also obtain a higher target action trigger rate.

(2) Results in the MCHT scenario. As shown in Tables 4 and 5, with the trigger threshold changes, the average round reward and the win rate is also gradually changes, which is similar to the results in the MCSPT scenario. In Fig. 9(a)-(d),we find that the attack effects produced by different target action attacks are different, but they all significantly reduce the team's average round reward and win rate, and a higher target action trigger rate can be obtained at the same time.

Ablation study To study the trigger action reward effect and relationship of the two c-MARL poisoning attacks, we perform an ablation study for the trigger action reward enhancement and the ensemble attack. The results are summarized in Tables 6 and 7.

(1) Results in the MCSPT scenario. In this experiment, we empirically set the trigger threshold to 3. Compared to poisoned policy without reward enhancement, the average round reward, the average round length and trigger action trigger rate of policy with reward enhancement has a slight reduction. Besides, the win rate have a slight improvement. The results show that the reward enhancement policy can strengthen the poisoning effect, and demonstrate its effectiveness for attacking multi-agent system. Compared with SNPA and TAPA in aspects of the average round reward, average round length, the win rate, and target action trigger rate, the ensemble attack has a better boosting for attack performance, which testifies that those two attack methods can attack at the same time with a stronger attack capacity.

(2) Results in the MCHT scenario. In this experiment, the trigger threshold is set to be 3. The results in the MCHT scenario are similar to that in the MCSPT scenario, and the growth ratio of the average round reward is less than in the MCSPT scenario. It further demonstrates that the policy of reward enhancement also helps the attack in this scenario. For the ensemble attack, the improvement effect of results is similar to that in the MCSPT scenario, which also testifies that the ensemble attack is scenario-adaptive.

(a) Different threshold, stay still

(b) threshold $\delta = 3$, Different Action

(c) threshold $\delta = 4$, Different Action

(d) threshold $\delta = 5$, Different Action

Fig. 8. The performance of TAPA in MCSPT on different threshold and target action.

Table 6
The performance of attacks with or without reward enhancement policy in both MCSPT and MCHT scenarios (100 rounds).

Scenarios	Reward Enhancement	Average round reward	Average round length	Win rate	Target action trigger rate
MCSPT	✗	-667.1 ± 30.7	93.28 ± 5.2	$9\% \pm 0$	$89.37\% \pm 1.3$
	√	-141.56 ± 19.8	98.61 ± 8.5	$4\% \pm 0$	$99.31\% \pm 0$
MCHT	✗	-1763.24 ± 113.5	79.64 ± 3.5	$46\% \pm 2$	$96.78\% \pm 0.4$
	√	-910.27 ± 52.4	86.18 ± 11.2	$32\% \pm 0.2$	$99.01\% \pm 0.2$

Table 7
The performance of the ensemble attack.

Scenarios	Average round reward	Average round length	Win rate	Target action trigger rate
MCSPT	-703.61 ± 20.5	97.4 ± 0.5	$2\% \pm 1$	$97.63\% \pm 0.2$
MCHT	-1389.87 ± 40.9	90.2 ± 1.2	$14\% \pm 1$	$98.53\% \pm 0.1$

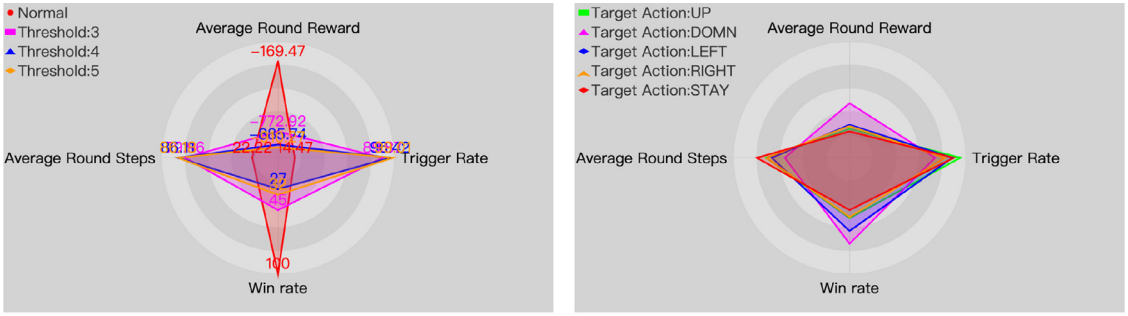
5. Related works

5.1. MARL and c-MARL

Single-agent reinforcement learning Single-agent RL (Henderson et al., 2018) concerns with how an intelligent agent ought to take actions in an environment in order to maximize the notion of cumulative reward. In other words, the agent optimizes a numerical performance by making decisions in stages. The decision-maker called an agent interacts with the environment of unknown dynamics in a trial-and-error fashion and occasionally receives feedback upon which the agent wants to improve. Generally, the standard formulation for such sequential decision-making is a Markov decision process (MDP) (Husic and Pande, 2018).

Multi-agent reinforcement learning Multi-agent reinforcement learning (MARL) is typically used for modeling the process of multiple agents participating in decision-making. Until now, numerous MARL methods have been proposed, such as, independent-Q-learning (IQL) (Shoham and Leyton-Brown, 2008; Tan, 1993; Tesauro, 2003; Zawadzki et al., 2014), counterfactual multi-agent policy gradients (COMA) (Foerster et al., 2018), QMIX (Rashid et al., 2020), multi-agent deep proximal policy optimization (MAPPO) (Yu et al., 2021), multi-agent deep deterministic policy gradient (MADDPG) (Lowe et al., 2017) and etc. These MARL algorithms have been successfully applied to robotic systems (Foerster et al., 2016; Gu et al., 2017), human-machine game (Lanctot et al., 2017; Leibo et al., 2017; Lowe et al., 2017), autonomous driving (Shalev-Shwartz et al., 2016), Internet advertising (Jin et al., 2018) and resource utilization (Perolat et al., 2017; Xi et al., 2018), etc. The process of multiple agents participating in decision-making is usually modeled as MARL.

Cooperative multi-agent reinforcement learning According to the way that the agent participates, MARL can be roughly divided into three categories, i.e., competitive MARL (Deka and Sycara, 2021; Ma et al., 2021), cooperative MARL (Cui and Zhang, 2021; Iranfar et al., 2021; Maxim and Caruntu, 2021; Panait and Luke, 2005) and competitive-cooperative MARL (Aotani et al., 2021; Vanneste et al., 2021). The c-MARL setting is that several agents attempt, through their interaction, to jointly solve tasks or to maximize utility. Specifically, in a c-MARL setting, the multiple agents conduct



(a) Different threshold, stay still

(b) threshold $\delta = 3$, Different Action

(c) threshold $\delta = 4$, Different Action

(d) threshold $\delta = 5$, Different Action

Fig. 9. The performance of TAPA in MCHT on different threshold and target action.

strategic collaborative learning as a team. Each agent maximizes the overall reward of the team by interacting with the environment and other agents, which can solve the sequential policy optimization problem of the multiple agents' decision-making in a conventional environment.

5.2. Poisoning attack

A poisoning attack occurs when the adversary can inject the poisoning data into the model's training pool, and hence get it to learn poisoned information. The most common result of a poisoning attack is that the model's boundary shifts in some designated way. Poisoning attack inserts hidden associations or triggers to the deep learning models to override correct inference such as classification (Xiang et al., 2021; Xue et al., 2020), and makes the system perform normally without triggering.

Poisoning attack implants a backdoor into the DL model in the way that the poisoned model learns both the sub-task chosen by the attacker and the main task. On the one hand, the poisoned model behaves normally as its clean counterpart model for input containing no trigger, making it impossible to distinguish the poisoned model from the clean model by solely checking the test accuracy with the test examples. On the other hand, the poisoned model is misdirected to perform the attacker's sub-task once the secret trigger is presented in the input.

There are two ways that the adversary can poison the RL setting. *(1) Dataset poisoning* (Li et al., 2019; Rakhsha et al., 2020). Dataset poisoning attack is a method that can corrupt a model. In this case, the adversary introduces incorrect or mislabeled data into the datasets. Alternatively, the adversary can change its behavior so that the data collected itself will be wrong. *(2) Algorithm poisoning* (Ma et al., 2019). In this type, the attacker takes advantage of the algorithm used to learn the model. The attacker often transfers learning where attackers teach an algorithm and then spread it to new RL algorithms using transfer learning.

5.3. Security issues of RL and MARL

For the multi-agent RL, Lin et al. (2020) was the first to study the robustness of c-MARL. They added adversarial perturbations to the victim's agent environment observations during the testing. This will reduce the overall performance of the team by taking the expected actions of the victim agent. Nisioti et al. (2021) proposed the RoM-*Q* method, which makes the worst-case selection of the agent to be attacked, and the action to be performed on the premise that the adversary knows the optimal *Q*-value function of multi-agent. Although these two methods have studied the robust security of the agent in the c-MARL scenario, they have not studied the security vulnerabilities that exist in the collaborative training process of the agent.

5.4. Defense methods of RL and MARL

With the widespread use of RL, we recently observe the use of robust optimization defense methods to improve the robustness of the RL model to resist attacks. Pinto et al. (2017) proposed a robust adversarial reinforcement learning defense method, an agent with a confrontation policy was added to enhance the policy of the target agent in the training process. Bravo and Mertikopoulos (2017) and Ogunmolu et al. (2018) respectively proposed an equilibrium principle and a maximum and minimum dynamic game framework for the zero-sum game problem.

In addition, for other learning tasks of RL, defense methods such as adversarial training and adversarial detection have also been used to strengthen the security of the model. For adversarial training security defense, both Kos and Song (2017) and Pattanaik et al. (2017) added fast gradient signal attack(FGSM) disturbances as adversarial examples to training examples, so that they can train the model together with normal examples to improve robustness. Similarly, Behzadan and Munir (2017) also utilized the FGSM perturbation, but they used a certain prob-

ability to generate adversarial examples for adversarial training. Behzadan and Hsu (2019a) improved the use rate of non-continuous anti-disturbance examples in anti-disturbance training by adjusting the sampling probability of normal examples and adversarial examples. Although the defense method of adversarial training can improve the perturbation attack on the adversarial examples participating in the training, but its generalization ability is limited and cannot effectively defend against the adversarial examples generated by other attack methods. For adversarial detection security defense, Havens et al. (2018) used meta-learning methods to detect the sub-strategies of the main agent and switched the sub-strategies once they are far away from the expected goal, thereby improving the effectiveness of the model policy. In order to enhance the robustness of the model, Lin et al. (2017) detected adversarial examples by comparing the target policy's action distribution difference between the predicted frame and the current frame. In (Behzadan and Hsu, 2019b), Behzadan et al. added a unique watermark (Uchida et al., 2017) to a specific state transition sequence to detect whether the policy has been tampered with, which improved the security application of the DRL model policy.

6. Conclusion

In this paper, two poisoning attacks aimed at the c-MARL setting are introduced, which can attack only one agent to affect the team's overall collaboration. Thus, the proposed attack methods pose a serious threat to the real-world multi-agent collaboration model. Extensive experiments have been conducted in two scenarios that verified SNPA and TAPA are effectiv and stealthy for attacking the c-MARL setting. Additionally, we find that the policy of maliciously modifying the state data or manipulating the victim agent can greatly affect the team's overall collaboration performance and achieve a low win rate. Besides, TAPA can also cause the victim agent to perform the expected target action, thereby disrupting the overall collaboration. Since multi-agent models are widely used in machine learning and robotic systems, such as autonomous-driving and transaction systems, studying the vulnerability of c-MARL is of great significance. SNPA and TAPA devote to fooling the victim model. However, with the increasing number of agents, the strategy effect of the victim agent is gradually weakened, and the effect of poisoning policy may diminish or even disappear.

The result shows that the c-MARL model has serious security risks, and its security issues still need to be resolved. There are two side of suggestions that are provided for future works: **Recommendations for Vulnerability Mining:** For cooperation strategy, both training policy and environment interaction should be considered for pollution, to reduce the impact of the imbalance of agents. **Suggestions for Defenders:** Researchers are advised to focus on the policy polymerization problem on c-MARL, which diminish the effect of the victim model by taking different weights.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Haibin Zheng: Conceptualization, Data curation, Formal analysis, Funding acquisition, Methodology, Resources, Supervision, Writing – review & editing. **Xiaohao Li:** Investigation, Methodology, Project administration, Resources, Software, Writing – original draft. **Jinyin Chen:** Investigation, Methodology, Software, Validation. **Jianfeng Dong:** Methodology, Software, Visualization, Writing – original draft. **Yan Zhang:** Conceptualization, Methodology. **Changting Lin:** Conceptualization, Methodology, Supervision, Writing – review & editing.

Data availability

Data will be made available on request.

Acknowledgments

This work was supported by NSFC (No. 62102363), NSFC (No. 62072406), CNKLSTISS (No. 61421110502), NSFC (Nos. U21B2001, 62103374), Key R&D Programs of Zhejiang Province (No. 2022C01018), the Zhejiang Provincial Natural Science Foundation (No. LQ21F020010).

References

Aotani, T., Kobayashi, T., Sugimoto, K., 2021. Bottom-up multi-agent reinforcement learning by reward shaping for cooperative-competitive tasks. Appl. Intell. 51 (7), 4434–4452.
Ashcraft, C., Karra, K., 2021. Poisoning deep reinforcement learning agents with in-distribution triggers. arXiv preprint arXiv:2106.07798.
Behzadan, V., Hsu, W., 2019a. Analysis and improvement of adversarial training in DQN agents with adversarially-guided exploration (age). arXiv preprint arXiv:1906.01119.
Behzadan, V., Hsu, W., 2019b. Sequential triggers for watermarking of deep reinforcement learning policies. arXiv preprint arXiv:1906.01126.
Behzadan, V., Munir, A., 2017. Whatever does not kill deep reinforcement learning, makes it stronger. arXiv preprint arXiv:1712.09344
Blas, H.S.S., Mendes, A.S., Encinas, F.G., Silva, L.A., González, G.V., 2021. A multi-agent system for data fusion techniques applied to the internet of things enabling physical rehabilitation monitoring. Appl. Sci. 11 (1), 331.
Böhmer, W., Kurin, V., Whiteson, S., 2020. Deep coordination graphs. In: International Conference on Machine Learning. PMLR, pp. 980–991.
Bravo, M., Mertikopoulos, P., 2017. On the robustness of learning in games with stochastically perturbed payoff observations. Games Econ. Behav. 103, 41–66.
Cui, H., Zhang, Z., 2021. A cooperative multi-agent reinforcement learning method based on coordination degree. IEEE Access 9, 123805–123814.
Deka, A., Sycara, K., 2021. Natural emergence of heterogeneous strategies in artificially intelligent competitive teams. In: International Conference on Swarm Intelligence. Springer, pp. 13–25.
Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., Whiteson, S., 2018. Counterfactual multi-agent policy gradients. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32.
Foerster, J. N., Assael, Y. M., De Freitas, N., Whiteson, S., 2016. Learning to communicate with deep multi-agent reinforcement learning. arXiv preprint arXiv:1605.06676.
Foley, H., Fowl, L., Goldstein, T., Taylor, G., 2022. Execute order 66: targeted data poisoning for reinforcement learning. arXiv preprint arXiv:2201.00762.
Gibert, D., Fredrikson, M., Mateu, C., Planes, J., Le, Q., 2022. Enhancing the insertion of NOP instructions to obfuscate malware via deep reinforcement learning. Comput. Secur. 113, 102543.
Gu, S., Holly, E., Lillicrap, T., Levine, S., 2017. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In: 2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE, pp. 3389–3396.
Guo, J., Li, A., Liu, C., 2022. Backdoor detection in reinforcement learning. arXiv preprint arXiv:2202.03609.
Hasselt, H.V., Guez, A., Silver, D., 2016. Deep reinforcement learning with double Q-learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30.
Havens, A. J., Jiang, Z., Sarkar, S., 2018. Online robust policy learning in the presence of unknown adversaries. arXiv preprint arXiv:1807.06064.
Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., Meger, D., 2018. Deep reinforcement learning that matters. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32.
Husic, B.E., Pande, V.S., 2018. Markov state models: from an art to a science. J. Am. Chem. Soc. 140 (7), 2386–2396.
Iranfar, A., Zapater, M., Atienza, D., 2021. Multiagent reinforcement learning for hyperparameter optimization of convolutional neural networks. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 41 (4), 1034–1047.
Jin, J., Song, C., Li, H., Gai, K., Wang, J., Zhang, W., 2018. Real-time bidding with multi-agent reinforcement learning in display advertising. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pp. 2193–2201.
Jinyin, C., Haibin, Z., Haibin, Z., Mengmeng, S., tianyu, D., Changting, L., Shouling, J., 2019. Invisible poisoning: highly stealthy targeted poisoning attack. In: International Conference on Information Security and Cryptology, pp. 173–198.
Kasseroller, K., Thaler, F., Payer, C., Štern, D., 2021. Collaborative multi-agent reinforcement learning for landmark localization using continuous action space.

In: International Conference on Information Processing in Medical Imaging. Springer, pp. 767–778.
Khalilpourazari, S., Hashemi Doulabi, H., 2022. Designing a hybrid reinforcement learning based algorithm with application in prediction of the COVID-19 pandemic in quebec. Ann. Oper. Res. 312 (2), 1261–1305.
Kim, D.K., Liu, M., Riemer, M.D., Sun, C., Abdulhai, M., Habibi, G., Lopez-Cot, S., Tesauro, G., How, J., 2021. A policy gradient algorithm for learning to learn in multiagent reinforcement learning. In: International Conference on Machine Learning. PMLR, pp. 5541–5550.
Kos, J., Song, D., 2017. Delving into adversarial attacks on deep policies. arXiv preprint arXiv:1705.06452.
Lanctot, M., Zambaldi, V., Gruslys, A., Lazaridou, A., Tuyls, K., Pérolat, J., Silver, D., Graepel, T., 2017. A unified game-theoretic approach to multiagent reinforcement learning. arXiv preprint arXiv:1711.00832
Lange, S., Riedmiller, M., Voigtländer, A., 2012. Autonomous reinforcement learning on raw visual input data in a real world application. In: The 2012 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. doi:10.1109/IJCNN.2012.625282.
Leibo, J. Z., Zambaldi, V., Lanctot, M., Marecki, J., Graepel, T., 2017. Multi-agent reinforcement learning in sequential social dilemmas. arXiv preprint arXiv:1702.03037.
Li, M., Sun, Y., Lu, H., Maharjan, S., Tian, Z., 2019. Deep reinforcement learning for partially observable data poisoning attack in crowdsensing systems. IEEE Internet Things J. 7 (7), 6266–6278.
Lin, J., Dzeparoska, K., Zhang, S.Q., Leon-Garcia, A., Papernot, N., 2020. On the robustness of cooperative multi-agent reinforcement learning. In: 2020 IEEE Security and Privacy Workshops (SPW). IEEE, pp. 62–68.
Lin, Y.-C., Liu, M.-Y., Sun, M., Huang, J.-B., 2017. Detecting adversarial attacks on neural network policies with visual foresight. arXiv preprint arXiv:1710.00814.
Liu, H., Wu, W., 2021. Online multi-agent reinforcement learning for decentralized inverter-based volt-var control. IEEE Trans. Smart Grid 12 (4), 2980–2990.
Liu, K., Dolan-Gavitt, B., Garg, S., 2018. Fine-pruning: defending against backdooring attacks on deep neural networks. In: International Symposium on Research in Attacks, Intrusions, and Defenses. Springer, pp. 273–294.
Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., Mordatch, I., 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. arXiv preprint arXiv:1706.02275.
Ma, Y., Shen, M., Zhao, Y., Li, Z., Tong, X., Zhang, Q., Wang, Z., 2021. Opponent portrait for multiagent reinforcement learning in competitive environment. Int. J. Intell. Syst. 36 (12), 7461–7474.
Ma, Y., Zhang, X., Sun, W., Zhu, J., 2019. Policy poisoning in batch reinforcement learning and control. Adv. Neural Inf. Process. Syst. 32, 1–11.
Van der Maaten, L., Hinton, G., 2008. Visualizing data using t-SNE. J. Mach. Learn. Res. 9 (11).
Maeda, R., Mimura, M., 2021. Automating post-exploitation with deep reinforcement learning. Comput. Secur. 100, 102108.
Mahajan, A., Samvelyan, M., Mao, L., Makoviychuk, V., Garg, A., Kossaifi, J., Whiteson, S., Zhu, Y., Anandkumar, A., 2021. Tesseract: tensorised actors for multi-agent reinforcement learning. In: International Conference on Machine Learning. PMLR, pp. 7301–7312.
Maxim, A., Caruntu, C.-F., 2021. A coalitional distributed model predictive control perspective for a cyber-physical multi-agent application. Sensors 21 (12), 4041.
Nisioti, E., Bloembergen, D., Kaisers, M., 2021. Robust multi-agent Q-learning in cooperative games with adversaries. AAAI.
Ogunmolu, O., Gans, N., Summers, T., 2018. Minimax iterative dynamic game: application to nonlinear robot control tasks. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, pp. 6919–6925.
Panait, L., Luke, S., 2005. Cooperative multi-agent learning: the state of the art. Auton. Agents Multi-Agent Syst. 11 (3), 387–434.
Pattanaik, A., Tang, Z., Liu, S., Bommannan, G., Chowdhary, G., 2017. Robust deep reinforcement learning with adversarial attacks. arXiv preprint arXiv:1712.03632.
Perolat, J., Leibo, J. Z., Zambaldi, V., Beattie, C., Tuyls, K., Graepel, T., 2017. A multi-agent reinforcement learning model of common-pool resource appropriation. arXiv preprint arXiv:1707.06600.
Perrusquía, A., Yu, W., Li, X., 2021. Multi-agent reinforcement learning for redundant robot control in task-space. Int. J. Mach. Learn. Cybern. 12 (1), 231–241.
Pinto, L., Davidson, J., Sukthankar, R., Gupta, A., 2017. Robust adversarial reinforcement learning. In: International Conference on Machine Learning. PMLR, pp. 2817–2826.
Rakhsha, A., Radanovic, G., Devidze, R., Zhu, X., Singla, A., 2020. Policy teaching via environment poisoning: training-time adversarial attacks against reinforcement learning. In: International Conference on Machine Learning. PMLR, pp. 7974–7984.
Rashid, T., Samvelyan, M., De Witt, C.S., Farquhar, G., Foerster, J.N., Whiteson, S., 2020. Monotonic value function factorisation for deep multi-agent reinforcement learning. J. Mach. Learn. Res. 21, 178-1.
Roderick M., MacGlashan J., Tellex S.. Implementing the deep Q-network. 2017. arXiv preprint arXiv:1711.07478.
Shalev-Shwartz, S., Shammah, S., Shashua, A., 2016. Safe, multi-agent, reinforcement learning for autonomous driving. arXiv preprint arXiv:1610.03295.
Shoham, Y., Leyton-Brown, K., 2008. Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations. Cambridge University Press.
Tan, M., 1993. Multi-agent reinforcement learning: independent vs. cooperative agents. In: Proceedings of the Tenth International Conference on Machine Learning, pp. 330–337.
Tesauro, G., 2003. Extending Q-learning to general adaptive multi-agent systems. Adv. Neural Inf. Process. Syst. 16, 871–878.
Uchida, Y., Nagai, Y., Sakazawa, S., Satoh, S., 2017. Embedding watermarks into deep neural networks. In: Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval, pp. 269–277.
Vanneste, A., Wijnsberghe, W.V., Vanneste, S., Mets, K., Mercelis, S., Latré, S., Hellinckx, P., 2021. Mixed cooperative-competitive communication using multi-agent reinforcement learning. In: International Conference on P2P, Parallel, Grid, Cloud and Internet Computing. Springer, pp. 197–206.
Xi, L., Chen, J., Huang, Y., Xu, Y., Liu, L., Zhou, Y., Li, Y., 2018. Smart generation control based on multi-agent reinforcement learning with the idea of the time tunnel. Energy 153, 977–987.
Xiang, Z., Miller, D.J., Kesidis, G., 2021. Reverse engineering imperceptible backdoor attacks on deep neural networks for detection and training set cleansing. Comput. Secur. 106, 102280.
Xie, Z., Xiang, Y., Li, Y., Zhao, S., Tong, E., Niu, W., Liu, J., Wang, J., 2021. Security analysis of poisoning attacks against multi-agent reinforcement learning. In: International Conference on Algorithms and Architectures for Parallel Processing. Springer, pp. 660–675.
Xue, M., He, C., Wang, J., Liu, W., 2020. LOPA: a linear offset based poisoning attack method against adaptive fingerprint authentication system. Comput. Secur. 99, 102046.
Yu, C., Velu, A., Vinitsky, E., Wang, Y., Bayen, A., Wu, Y., 2021. The surprising effectiveness of MAPPO in cooperative, multi-agent games. arXiv preprint arXiv:2103.01955.
Zawadzki, E., Lipson, A., Leyton-Brown, K., 2014. Empirically evaluating multiagent learning algorithms. arXiv preprint arXiv:1401.8074.
Zemzem, W., Tagina, M., 2018. Cooperative multi-agent systems using distributed reinforcement learning techniques. Procedia Comput. Sci. 126, 517–526.

Haibin Zheng received B.S. and Ph.D. degrees from Zhejiang University of Technology, Hangzhou, China, in 2017 and 2022, respectively. He is currently a university lecturer at the Institute of Cyberspace Security, Zhejiang University of Technology. His research interests include deep learning and artificial intelligence security.

Xiaohao Li is a postgraduate student at the College of Information Engineering, Zhejiang University of Technology. His research interests include deep reinforcement learning, artificial intelligence, deep learning.

Jinyin Chen received B.S. and Ph.D. degrees from Zhejiang University of Technology, Hangzhou, China, in 2004 and 2009, respectively. She is currently a Professor with the Zhejiang University of Technology, Hangzhou, China. Her research interests include artificial intelligence security, graph data mining and evolutionary computing.

Jianfeng Dong received the B.E. degree in software engineering from Zhejiang University of Technology in 2009, and the Ph.D. degree in computer science from Zhejiang University in 2018, all in Hangzhou, China. He is currently a Research Professor at the College of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou, China. His research interests include multimedia understanding, retrieval and recommendation. He was awarded the ACM Multimedia Grand Challenge Award in 2016. He has won a number of international competitions including the TRECVID 2016, 2017, 2018 Video-to-Text (VTT) Matching and Ranking task, the MSR Bing Image Retrieval Challenge at ACM Multimedia 2015, and so on.

Yan Zhang received the master degree from ZheJiang University of Technology, HangZhou, China, in 2021. Her main research methods are artificial intelligence security, computer vision.

Changting Lin received the Ph.D. degree in computer science from Zhejiang University in 2018. He is currently an Associate Research Professor at Binjiang Institute of Zhejiang University, China. His research interests include blockchain technology, artificial intelligence, network, and security.