Advanced Persistent Threat in Cooperative Multi-Agent Reinforcement Learning

Anonymous Author(s)
Submission Id: 169

ABSTRACT

Cooperative Multi-Agent Reinforcement Learning (c-MARL) enables a team of agents to determine the global optimal policy that maximizes the sum of their accumulated rewards. In this paper, we investigate the robustness of c-MARL to a novel advanced persistent threat (APT). In this APT, we target and weaponize one agent, termed the *compromised agent*, to create unnatural observations that are adversarial for its team. Then, we adopt an adversarial example method from deep learning, FGSM, to enhance the attack with a detection evasion technique. The goal is to lure the compromised agent to follow an adversarial policy that pushes activations of its cooperative agents' policy networks off distribution while avoiding detection. This paper shows mathematically the exploitation steps of such adversarial policy in the centralized-learning and decentralized-execution paradigm of c-MARL. We also empirically demonstrate the susceptibility of the state-of-the-art c-MARL algorithms; MADDPG and QMIX to our APT using 4 attack strategies in three environments in white and black box settings. By targeting a single agent, our attacks have highly negative impact on the overall team reward in all environments, reducing it by at least 33% and at most 89.6% with an evasion rate of at least 16% and at most 36%. We uploaded our code and datasets as supplementary material for ensuring the reproducibility of our research findings.

KEYWORDS

MARL, adversarial, FGSM, APT, robustness

1 INTRODUCTION

Advances in single-agent reinforcement learning (RL) algorithms sparked new interest in cooperative Multi-Agent Reinforcement Learning (c-MARL). Several c-MARL training algorithms have been developed in response to demand in such areas as cyber-physical systems, sensor/communication networks, and social science. However, the robustness of c-MARL algorithms has not been investigated under adversarial policies except in [11]. Adversarial attacks on c-MARL are different from those on an individual RL agents [5, 7, 9] in several ways. First, each agent in c-MARL interacts and modifies the state of the environment not only for itself but also for its cooperative agents. For an episode of L steps, an adversary has $2^L * N$ choices of attacking or not attacking at least one agent at each time step in a system with N agents which significantly amplifies the attack surface for c-MARL compared to the individual RL agent. Second, an adversary to c-MARL has different goals, such as reducing the final rewards of the whole team or maliciously using some agents' actions to lure other agents to dangerous states. We call this vulnerability *compromised agent vulnerability*, which is different from adversarial attacks against an individual RL agent that aim to directly minimize the agent's reward. The compromised agent vulnerability in c-MARL has been only investigated in one recent work [11] in a white-box settings against QMIX algorithm using a modified version of the adversarial example attack, JSMA [22], in one environment.

The compromised agent vulnerability is relevant to the robustness of the multi-agent domains such as autonomous driving in Platoons [16], negotiation [19], and automated scalping trading [8]. In those settings, agents work in environments populated by other agents, including humans, which increases the number of potential adversaries against the system. In such domains, it is not usually feasible for the adversary to directly change the victim policy's input via adversarial examples, but instead can only modify the victim agent's observations via its own actions. For example, in platoons of autonomous vehicles, pedestrians and other drivers cannot add noise to arbitrary pixels, or make a building disappear but they can take adversarial actions that negatively affect the camera image, but only in a physically realistic fashion. Similarly, in financial trading, an adversary can send orders to an exchange which will appear in the victim's market data feed, but the attacker cannot modify observations of a third party's orders.

In this paper, we reverse engineer the c-MARL algorithms in the centralized-learning and decentralized-execution paradigm to exploit the compromised agent vulnerability. We also empirically analyze the impact of this vulnerability on the robustness of two of the state-of-the-art algorithms in 3 different environments. Our contributions are listed as follows:

(1) We show the feasibility of the compromised agent vulnerability by reverse-engineering the centralized-learning decentralized-execution paradigm of c-MARL.
(2) We design 4 attack strategies to demonstrate the impact of the compromised agent vulnerability on the robustness of c-MARL algorithms in white and black box settings.
(3) We develop anomalous behavior detection models using different machine learning techniques to detect the compromised agent in c-MARL.
(4) We extend our attack strategies with an evasion technique to make them APTs that are hard to detect while causing destructive impact.
(5) We thoroughly analyze the robustness of the lead c-MARL algorithms MADDPG [14] and QMIX [17] under our attacks and their APTs counterparts in three different environments in cooperative and competitive settings. Our compromised agent brings down the team reward rate in all environments by at least 33% and at most 89.6% with an evasion rate of at least 16% and at most 36%.

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Auckland, New Zealand. © 2022 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

2 RELATED WORK

Previous work has shown the vulnerability of individual RL agents to adversarial attacks, in which the adversary perturbs the agent's observation to degrade its performance [7]. Other attacks reduce the number of adversarial examples needed to decrease the agent's reward [12] or trigger misbehavior of the agent [25]. None of this work studies c-MARL, and the effects of cooperation on the success of the attacks. The closest work to ours is [11] which proposes a two-step attack against QMIX [17] with the objective of reducing the total team reward by perturbing the observation of a single agent. Their work extends an existing adversarial example method JSMA [22] to create d-JSMA which is more suitable for attacking an RL model with a low-dimensional feature space. They focus on white-box settings to launch their attack by assuming the knowledge of the team reward function. Our work differs from their approach by using a physically realistic attack model that does not depend on directly modifying the agents' observations with adversarial examples. Instead, we weaponize one agent to follow an adversarial policy that pushes activations of its cooperative agents' policy networks off distribution. We also conduct our attacks in both white and black box settings against QMIX and MADDPG in 3 environments. Furthermore, we implement a detection mechanism and enhance our attacks to evade the detection. Another work that is close to ours is Gleave et al. [5]. However, the focus of this work is on attacks in the competitive multi-agent setting whereas we consider cooperative teams of agents in a both fully-cooperative and competitive environments.

3 PRELIMINARIES

3.1 c-MARL

In this paper, we model the c-MARL system using stochastic games [18]. For an n-agent stochastic game, we define a tuple:

$$G = \left(S, A^1, ..., A^n, r^1, ..., r^n, T, \gamma\right) \tag{1}$$

where S denotes the state space, A^i and r^i are the action space and the reward function for agent $i \in 1, ..., n$ respectively. γ is the discount factor for future rewards, and T is a joint state transition function $T : S \times A_1 \times A_2 .. \times A_n \rightarrow \triangle(S)$ where $\triangle(S)$ is a probability distribution on S. Agent i chooses its action $a^i \in A^i$ according to its policy $\pi^i_{\theta^i}(a^i|s)$ parameterized by θ^i conditioning on some given state $s \in S$. The collection of all agents' policies π_θ is called the joint policy and θ represents the joint parameter. For convenience, we interpret the joint policy from the perspective of agent i as:

$$\pi_\theta = (\pi^i_{\theta^i}(a^i|s)\pi^{-i}_{\theta^{-i}}(a^{-i}|s)) \tag{2}$$

where $a^{-i} = (a^j)_{j \neq i}, \theta^{-i} = (\theta^j)_{j \neq i}$, and $\pi^{-i}_{\theta^{-i}}(a^{-i}|s)$ is a compact representation of the joint policy of all complementary agents of i [13]. At each stage of the game, actions are taken simultaneously.

The centralized learning and decentralized execution paradigm of MARL has been followed by the major c-MARL algorithms including MADDPG [14], COMA [2], MF-AC [24], Multi-Agent Soft-Q [20], LOLA [3], and Q-Mix [17].The focus of this paper is on attacking MADDPG [14] and Q-Mix [17], the leading algorithms in this paradigm.

MADDPG extends DDPG into a multi-agent policy gradient algorithm in which decentralized agents learn based on the observations and actions of all agents using a centralized unit called the critic. Then, each agent has a local policy (called an actor) that only uses local information (i.e. its own observations) at execution time. The critic is augmented with extra information about the policies of other agents, while the actor only has access to its local information. After training is completed, only the local actors are used at execution phase, acting in a decentralized manner.

Q-MIX is a multi-agent Deep Q-Learning (DQN) algorithm based on Q-Learning with state-of-the-art performance on the StarCraft II Multi-Agent Challenge [17]. QMIX finds the optimal joint action value function using a monotonic mixing function of per-agent utilities. In QMIX, each agent uses a recurrent-DQN network to estimate the action values based on their partial observation. DQN uses experience replay and a target network to improve stability and convergence of RL agent's training. To maximize the total team reward, QMIX estimates the joint action values through a mixing network that takes in each agent's selected action Q-value (i.e., action with max Q-value) and the current state to estimate the total team reward.

3.2 Anomalous Behavior Detection in c-MARL

For detecting compromised agents, we developed different anomalous behavior detection models using a 4 layer Dense deep learning, Random Forest, SVM classifier, K-nearest Neighbors, a 3 layer binary LSTM, predictive LSTM similar to the model in [1], and an ensemble of the binary LSTM and the predictive LSTM models. We trained those models on clean datasets with 220,000 samples of normal datapoints generated from each environment with fully cooperative agents. We found that the ensemble model achieved the highest precision and recall across all models when tested on datasets generated by our attacks. The details of the models and their detection results are in Appendix A.1.

The ensemble LSTM model (Figure 1 in Appendix A.1) predicts the sequence of actions expected to be taken by the compromised agent m at the next h time steps $t + h$ based on the actions taken by the benign agents $-m$, the actions taken by m at the last $t - h$ time steps, and the states $s_t, s_{t-1}, ..s_{t-h}$. Let's assume the prediction function for the ensemble LSTM is $\overrightarrow{y} = \omega(x)$ such that y is the prediction output and x represents the input to the network. $\overrightarrow{y}$ is a vector of the predicted next h actions for agents m; $[a^m_t, .., a^m_{t+h}]$ and the input x is three vectors of historical information including; a vector for the previous h actions taken by agents m; $[a^m_{t-h}, ..., a^m_{t-1}]$, a vector for the previous h actions taken by agents $-m$; $[a^{-m}_{t-h}, ..., a^{-m}_{t-1}]$, and a vector of previous states $[s_{t-h}, ..., s_{t-1}]$. Then, the LSTM prediction function becomes:

$$\overrightarrow{a}^m_{t,t+h} = \omega^m \left([a^m_{t-h}, .., a^m_t], [a^{-m}_{t-h}, .., a^{-m}_t], [s_{t-h}, .., s_t]\right) \tag{3}$$

Using the results from Eq.3, we compute the error vector $\rightarrow e$ based on Mahalanobis' distance [15] between the predicted and the observed vectors of actions as:

$$\overrightarrow{e}_{t,t+h} = \overrightarrow{a}^m_{t,t+h_{obs}} - \overrightarrow{a}^m_{t,t+h_{pred}} \tag{4}$$

Using a predefined threshold σ, we compute how likely the error vector $\overrightarrow{e}$ represents an anomaly if it is larger than σ.

4 APPROACH

4.1 Problem Formulation: Advanced Persistent Threat in c-MARL

We formulate the compromised agent APT as an optimization problem to simultaneously accomplish two main objectives; **Obj 1**: optimizing its attack strategies by selecting strong destructive actions to the c-MARL within a time budget, and **Obj 2**: evading the anomalous behavior detection model (Eq.4). These two objectives are often in conflict such that achieving stronger sabotage in Obj 1 typically increases the detection rate of anomalies in Obj 2 and vice versa.

We assume all agents, including the compromised one, follow fixed policies corresponding to the common case of a pretrained model, deployed with static weights. This model holds particularly well for safety-critical systems, where it is a standard practice to validate a model, then freeze it, to ensure that it does not develop any new problems due to training [5]. Since the agents' policies are held fixed, the Markov game G (Eq.1) reduces to a single-player MDP, $G_m = (S, A_m, T_m, R_m)$ that the adversary must solve to generate a new policy by which the compromised agent m will achieve the adversary's goal of attacking the other agents $-m$ while evading the detection model.

The optimization problem of APT in c-MARL is defined as follows:

$$\begin{aligned} &\max \sum_{t=0}^{t=T} \mathbf{KL}\left(p(a_t^{-m}|a_t^m, s_t)||p(a_t^{-m}|a_t^{*m}, s_t\right) \text{ subject to} \\ &\min[\overrightarrow{e}^{*m}_{t,t+h}] \forall 1 < h < T, h \bmod \mathrm{k} = 0 \end{aligned} \tag{5}$$

where a^{*m} represents the adversarial actions generated by the adversarial policies for agent m. This optimization maximizes the KL-divergence between the conditional policy of $-m$ on the action a^m at time t and the same conditional policy if agent m deviates from its policy and takes an adversarial action a^{*m}. The adversary can then intervene on a_t^m by replacing it with the action, a_t^{*m} which will be used to compute the next action of $-m$, $p(a_{t+1}^{-m}|a_t^{*m}, s_t^m)$ pushing the activations of their policy networks off distribution. The detection model in Eq.4 ($\overrightarrow{e}^{*m}_{t,t+h}$)) can run prediction on each agent every time it performs a new action. However, this timely prediction can be potentially very expensive when the number of agents increases. Thus, we assume that Eq.4 only runs detection every time the agent takes k new actions. This makes h the earliest interval timestep at which the compromised agent m is detected.

Practically, we solve the optimization problem in Eq.5 in 2-steps. First, by finding the adversarial actions a^{*m} for the compromised agent following one of the proposed attack strategies (Sec.4.3) to maximize the KL-divergence in Eq.5. Then, to evade the detection model, we extend those attacks by adopting the adversarial example method FGSM [6] from the deep learning field to add strategic perturbations to the compromised agent's observations reported to the detection model Eq.4.

4.2 Exploitation of Compromised Agent Vulnerability

We now show the steps of exploiting the compromised agent vulnerability in the centralized-learning and decentralized-execution c-MARL algorithms. The goal is to reverse engineering the c-MARL

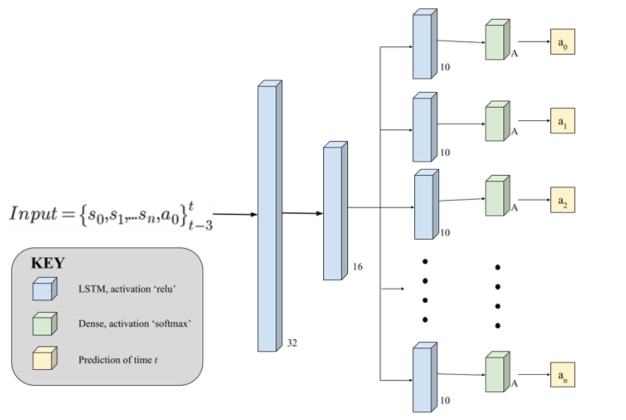


Figure 1: The architecture of the deep learning based behavioral cloning.

learning algorithm by correlating the learning process of m policy with the learning of the conditional policies of $-m$. First, we collect state-action pairs by observing the c-MARL agents during reconnaissance as a trajectory $\tau = [(s_1, a_1^m, a_1^{-m}), ..., (s_t, a_t^m, a_t^{-m})]$. We use τ to train Adversarial Inverse Reinforcement Learning (AIRL) [4] to discover the hidden reward function behind agent m's behavior $r^m(s_t, a_t^m, a_t^{-m}, s_{t+1})$ where we consider the actions taken by other agents a_t^{-m} as part of the environment state.

The second step is to reverse engineer the joint policy in Eq.2 to approximate the agents' policies. The joint policy in Eq.2 can be reformulated as [21, 23]:

$$\begin{aligned} \pi_\theta(a^m, a^{-m}|s) &= \underbrace{\pi^m_{\theta^m}(a^m|s)\pi^{-m}_{\theta^{-m}}(a^{-m}|s, a^m)}_{\text{Compromised agent's perspective}} \\ &= \underbrace{\pi^{-m}_{\theta^{-m}}(a^{-m}|s)\pi^m_{\theta^m}(a^m|s, a^{-m})}_{\text{other agents' perspective}} \end{aligned} \tag{6}$$

From the perspective of the compromised agent m, the first equality in Eq.6 indicates that the joint policy can be essentially decomposed into two parts; the agent m policy and the conditional policies of agents $-m$. The conditional part of the first equality $\pi^{-m}_{\theta^{-m}}(a^{-m}|s, a^m)$ represents what actions would be taken by victim agents $-m$ given the fact that they know the current state of the environment and agent m's action; that is, what the compromised agent believes the other agents might think about it, based on their original policy. In the white-box settings, the adversary has direct access to the agents' policies and the ground truth of the state transition function. In the black-box setting, the adversary needs to develop an approximation for the transition function and the conditional policies of the victims $\pi^{-m}_{\theta^{-m}}(a^{-m}|s, a^m)$ via the behavioral cloning model in Figure 1. This model is trained by minimizing the loss function $L(a, \pi_\theta)$ to approximate each agent's policy $\rho^{-m}_{\phi^{-m}}(a^{-m}|s, a^m)$. The supervised learning model for the state transition function is similar to the behavioral cloning model with multiple heads but with a different learnable parameters Φ (more details about behavioral cloning and state transition models given in Appendix B):

$$T_\Phi = p(s_{t+1}|s_t, a^1, ..., a^n) \tag{7}$$

By approximating the actual policies of the victims $-m$, the learning task for the compromised agent m can be formulated as

$$\arg\max_{\theta^m,\phi^{-m}} \left(\pi^m_{\theta^m}(a^m|s)\rho^{-m}_{\phi^{-m}}(a^{-m}|s,a^m)\right) \tag{8}$$

With the learning protocol in Eq.8, the adversary can learn the compromised agent m's policy and approximate the conditional policy of the victim agents $-m$ given m's actions using the probabilistic reinforcement learning [10]. We first derive the probability of τ being observed during the reconnaissance phase as

$$p(\tau) = \left[p(s_1)\prod_{t=1}^{T} p(s_{t+1}|s_t,a^m_t,a^{-m}_t)\right] \exp\left(\sum_{t=1}^{T} r^m(s_t,a^m_t,a^{-m}_t)\right) \tag{9}$$

Recall, we used AIRL [4] to discover r^m using the collected τ. The goal then becomes about finding the best approximation of $\pi^m_{\theta^m}(a^m_t|s_t)\rho^{-m}_{\phi^{-m}}(a^{-m}_t|s_t,a^m_t)$ to maximize Eq.8 such that the induced trajectory distribution $\hat{p}(\tau)$ can match the ground-truth of trajectory probability $p(\tau)$:

$$\hat{p}(\tau) = p(s_1)\prod_{t=1}^{T} p(s_{t+1}|s_t,a^m_t,a^{-m}_t)\pi^m_{\theta^m}(a^m_t|s_t)\rho^{-m}_{\phi^{-m}}(a^{-m}_t|s_t,a^m_t) \tag{10}$$

We can now optimize the approximated policies of the victim agents $-m$ by minimizing the KL-divergence between Eq.9 and Eq.10 as

$$\begin{aligned} KL(\hat{p}(\tau)||p(\tau)) &= -\mathbb{E}_{\tau\ \hat{p}(\tau)}[logp(\tau) - log\hat{p}(\tau)] \\ &= -\sum_{t=1}^{t=T} E_{\tau\ \hat{p}(\tau)}\left[r^m(s_t,a^m_t,a^{-m}_t)\right. \\ &\quad \left. +H\left(\pi^m_{\theta^m}(a^m_t|s_t)\rho^{-m}_{\phi^{-m}}(a^{-m}_t|s_t,a^m_t)\right)\right] \end{aligned} \tag{11}$$

where H is the conditional entropy on the joint policy that potentially promotes the exploration for both the malicious agent m's best action and the victims' conditional policies. Minimizing Eq.11 gives us the optimal Q-function for agent m as (Theorem 1 in [21]):

$$Q^m_{\pi_\theta} = log\int_{a^{-m}} \mathbb{E}(Q^m_{\pi_\theta}(s,a^m,a^{-m}))da^{-m} \tag{12}$$

And the corresponding $-m$ conditional policy becomes:

$$\rho^{-m}_{\phi^{-m}}(a^{-m}_t|s_t,a^m_t) = \frac{1}{Z}\mathbb{E}(Q^m_{\pi_\theta}(s,a^m,a^{-m}) - Q^m_{\pi_\theta}(s,a^m)) \tag{13}$$

To solve Eq.13, we maintain two Q-functions and iteratively update them using the learned reward function for agent m and the observations in τ as the ground-truth. Eq.12 approximates the original policy of agent m while considering the approximated conditional policies of agents $-m$ in Eq.13. The overall steps of exploiting the compromised agent vulnerability are shown in Figure 2. For the proof of Eq.12 and Eq. 13, please check [21].

4.3 Attack Strategies

In Figure 2, we still need to generate the adversarial policies for the compromised agent to choose a^{m*}_t. In this section, we introduce our attack strategies to generate those policies under two

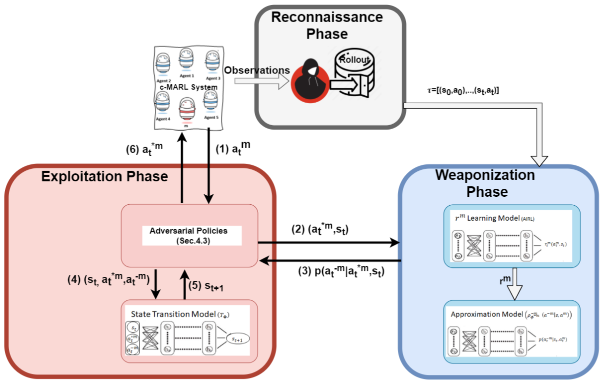


Figure 2: The steps of exploiting the compromised agent vulnerability

different categories: attacks based on the compromised agent's self-destruction strategies; and attacks based on destructive strategies of other agents' objectives

4.3.1 *Self-Destruction Adversarial Policies .*

This category focuses on minimizing the compromised agent's individual reward which indirectly reducing the team reward of the c-MARL.

Randomly-Timed Attack: At a certain percentage of timesteps, the adversary changes the compromised agent's action into a^{*m}_t based on a random off-distribution policy.

Strategically-Timed Attack: This attack strategically selects a subset of time steps to change the compromised agent's actions. We first calculate the c-function [12] as:

$$c(s_t) = \max_{a_t}(\pi_m(s_t,a_t)) - \min_{a_t}(\pi_m(s_t,a_t))$$

and launch the attack if $c > b$, where b is a chosen threshold that indicates the desired attacking rate. The idea behind this attack is that the adversary chooses to alter the compromised agent's action only when it strongly prefers a specific action (the action has a relative high probability), which means that it is critical to perform that action or its accumulated reward will be reduced.

4.3.2 *Adversarial Policies for Destructing Other Agents.*

The other method to attack c-MARL algorithms is to sabotage the objectives of the other agents in the system using the compromised agent actions which directly pushes activations of c-MARL agents' policy networks off-distribution.

Counterfactual Reasoning Based Attack: This attack predicts the compromised agent's counterfactual reasoning process about how its actions will affect the other agents and then postulates actions that would enable it to achieve maximal destruction to the system. We design a reward function for an RL agent to generate a long-term policy that replaces the original actions $a^m_t, ..., a^m_{t+l}$ to the best set of adversarial actions $a^{*m}_t, .., a^{*m}_{t+l}$. The RL agent would choose certain time steps to attack instead of attacking each time step based on how much divergence the attack is expected to produce. To do so, the reward function is crafted as

$$r_{att} = \sum_{t}^{l} \gamma^t \, \mathbf{KL} \left(p(a_t^{-m} | a_t^m, s_t) \; || \; p(a_t^{-m} | a_t^{*m}, s_t) \right) \tag{14}$$

where a^{*m} represents the set of counterfactual actions to the action a^m generated by the original policy of agent m; $\pi^m_{\theta^m}$. γ is a discounted factor, the attack starts at time step t, and l is the number of steps into the future when the attacker wants to the event to take place. Eq.14 generates a combination of actions $a_t^{*m}, ..., a_{t+l}^{*m}$ for which the success rate is the highest possible. We train this adversarial policy using DDPG.

Zero-Sum Attack: We train another RL agent to learn an adversarial policy minimizing the global reward of the c-MARL system. As mentioned before, the policies of all non-compromised agents $-m$ are fixed, hence from the compromised agent's m perspective, they are considered as part of the environment. Thus, we can formulate this attack as a single agent RL problem to minimize the cumulative reward for the team as:

$$\underset{\theta^m}{\operatorname{argmin}} \sum_{t=1}^{\infty} \gamma^t r_t(s_t, a_t^{m*}, a_t^{-m}, s_{t+1}), \tag{15}$$

where a_t^{m*} is the compromised agent's action at time step t, θ^m parameterizes the compromised agent's policy, and r_t is the global reward recoverd by [4]. With (a_t^{m*}, a_t^{-m}) sample from $(\pi^m_{\theta^m}, \pi^{-m}_{\theta^{-m}})$ in the white box setting. In the black box setting, we use sample from the approximated policies derived in Eq.12 and Eq.13. With Eq.15, we train an adversarial policy that selects actions for the compromised agent that will minimize the reward r for the c-MARL team. In our implementation, we use DDPG to train this adversarial policy.

4.4 Evasion Technique

To increase the stealthiness of our attacks and satisfy **Obj2**, we design an evasion technique adopting the fast gradient sign method (FGSM) from the deep learning field. We assume the adversary has no knowledge of the ensemble LSTM model's parameters (Eq.4) in an attempt to show the true implications of this technique. Recall that the FGSM equation [6] is $x_{\text{adversarial}} = \epsilon * sign(\Delta_x J(\Theta, x, y))$. In our design of FGSM for the evasion, we use FGSM equation to add an imperceptibly small vector of perturbation ϵ whose elements are equal to the sign of the elements of the gradient of the objective function of the detection model ω. After generating the adversarial action (a^m) using the aforementioned adversarial policies, we use FGSM to perturb the compromised agent's state ($x = s^m$). y is the output label (anomalous action or normal action), and Θ is the detection model's parameters. More details about this technique given in Appendix A.2.

5 EXPERIMENTS

We evaluate the robustness of c-MARL algorithm MADDPG under our attacks in both the white and black box settings in 2 particle environments [14]: cooperative navigation and physical deception with variant attacking rates. In the cooperative navigation, N cooperative agents must cover L landmarks, and the agents must learn to reach separate landmarks, without communicating their observations to each other. In our experiments, we use $N = L = 3$. In the physical deception environment, N cooperative agents try to fool one adversarial agent. There are L total landmarks, with one being the 'target' landmark and only the cooperative agents know which landmark is the target one. The adversary must try to infer and reach the target landmark from the cooperative agents' positioning, and the cooperative agents must try to deceive the adversary by spacing out. The cooperative agents are rewarded as long a single member of their team reaches the target landmark. We use $N = L = 2$.

To test the robustness of QMIX algorithm against our attacks, we use StarCraft Multi-Agent Challenge (SMAC) [17]. StarCraft II is a real-time strategy game where two teams of agents can fight against each other. We use the "3m" SMAC map, which employs three "Marine" units on each team. In this scenario, a team wins by shooting the enemy team enough to drain their health. Our team consists of three cooperative agents working together to defeat the three enemy marines, which use fixed policies. For our attacks, we control one of the cooperative marines, and alter its actions using the aforementioned attack strategies (Sec.4.3).

In all environments, the adversary wishes to choose a set of actions for the compromised agent, $A^* = [a_1^*, \ldots, a_T^*]$, where T is the episode length, such that:

$$R_{team}([s_1, .., s_N], A^*) <= R_{team}([s_1, .., s_N], A)$$

where R_{team} represents the reward for the cooperative team. Team reward is a direct measurement of the c-MARL quality; a successful attack should be able to decrease it. Beside the team reward, we use different environment-specific metrics to measure the attack success rate.

5.1 Results and Discussion

5.1.1 Adversarial Policies and Team Reward. To learn an adversarial policy for the compromised agent, we used the attack strategies explained in Sec.4.3 before using the evasion technique. To evaluate the performance of each policy, we directly change the actions of the compromised agent based on the output of the adversarial policy. We ran each attack with different attack episode lengths starting a 0% and moving to 100% by increments of 25%. We presented the team average reward for victim agents in Figure 3. From the results, as the attack rate increases, the team reward decreases, indicating that each attack was successful at harming the robustness of c-MARL algorithms. Strategically-timed attack has the highest negative impact on team reward when the attack rate is higher. However, the Zero-sum and counterfactual reasoning based attacks perform better with less attack rates in the cooperative navigation environments. In the physical deception environment, strategically-timed attack achieved 86.6% and 85% reward drop for 100% attacking rate in white-box and black-box settings respectively. In the cooperative navigation, strategically-timed attack achieved a reward drop of 33.1% and 33.0% for white-box and black-box settings respectively. In the StarCraft II environment, the strategically-timed attack achieved a reward drop of 84% and 89.6% in the white-box and the black-box settings respectively. The strategically-timed attack is the strongest because it attacks only at impactful timesteps

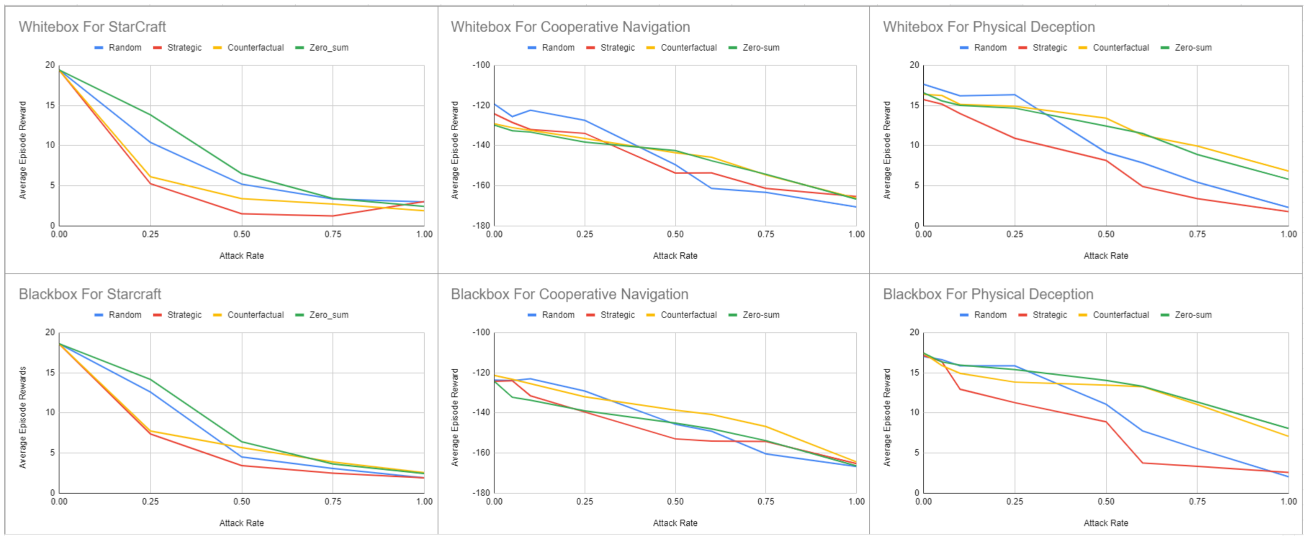


Figure 3: The average episode rewards in 2 particle multi-agent environments (MADDPG) and StarCraft2 (QMIX) under our attacks as a function of the attacking rate

Table 1: The average number of occupied landmarks in the cooperative navigation, the average distance between the cooperative agents and the target landmark in physical deception [14], and the win rate for the cooperative team in StarCraft II [17] for 25%, 50%, 75%, and 100% attack rates across all attack methods in both white and black box settings.

MADDPG: Cooperative Navigation (occupied landmarks)								
	White Box				**Black Box**			
Attack Rate	Random	Timed	Counterfactual	Zero-sum	Random	Timed	Counterfactual	Zero-sum
0%	1.611	1.611	1.611	1.611	1.611	1.611	1.611	1.611
25%	1.311	1.308	1.007	1.102	1.325	1.333	1.150	1.09
50%	0.958	0.955	0.786	0.868	1.058	1.048	1.002	0.847
75%	0.695	0.711	0.601	0.722	0.860	0.815	0.811	0.730
100%	0.502	0.562	0.489	0.573	0.502	0.688	0.547	0.519
MADDPG:Physical Deception (average distance)								
	White Box				**Black Box**			
Attack Rate	Random	Timed	Counterfactual	Zero-sum	Random	Timed	Counterfactual	Zero-sum
0%	0.163	0.163	0.163	0.163	0.163	0.163	0.163	0.163
25%	0.225	0.418	0.196	0.188	0.200	0.343	0.182	0.175
50%	0.326	0.470	0.211	0.277	0.324	0.499	0.201	0.2
75%	0.515	0.607	0.264	0.297	0.530	0.614	0.235	0.243
100%	0.540	0.688	0.427	0.423	0.654	0.680	0.412	0.356
QMIX: StarCraft II (win rate)								
	White Box				**Black Box**			
Attack Rate	Random	Timed	Counterfactual	Zero-sum	Random	Timed	Counterfactual	Zero-sum
0%	0.955	0.955	0.955	0.955	0.955	0.955	0.955	0.955
25%	0.32	0.18	0.190	0.210	0.475	0.135	0.135	0.528
50%	0.065	0.0	0.060	0.050	0.015	0.0	0.080	0.144
75%	0.01	0.0	0.0	0.000	0.0	0.0	0.030	0.04
100%	0.0	0.0	0.0	0.000	0.0	0.0	0.0	0.0

as opposed to the randomly-timed attack which attacks at random time steps during the episode.

The counterfactual reasoning-based attack is able to achieve similar level of performance with strategically-timed attack in StarCraft. In Cooperative Navigation, both the counterfactual and Zero-sum achieve similar performance as the strategically timed attack especially with low attack rate. We believe since those environments are purely cooperative environments, the adversarial policies trained to directly sabotage other agents' policies can better diminish the robustness of the algorithms in cooperative settings. However, in physical deception environment that naturally involves an enemy, counterfactual attack and Zero-sum attack have the least impact on the team reward. On the other hand, the counterfactual reasoning based attack and Zero-sum attack both were the least detectable attacks among all tested attacks in all environments as we will show in the defense section which makes them more powerful in the presence of anomaly detection.

Table 2: The detection results of the attacks using the ensemble LSTM model (*precision%|recall%*) in the cooperative navigation and physical deception environments across 25%, 50%,75%, and 100% respectively in white-box settings. The detection results for the black-box settings are given in Table 1 in Appendix A.1

MADDPG:Cooperative Navigation							
Random		Timed		Counterfactual		Zero-sum	
60%	96%	64%	98%	29%	79%	37%	83%
85%	98.5%	85.5%	98%	56%	88%	61%	88%
94.5%	99%	99%	95%	75%	82%	79%	88%
100%	98%	100%	99%	100%	86%	100%	91%
MADDPG:Physical Deception							
Random		Timed		Counterfactual		Zero-sum	
51%	96%	54%	97%	28%	80%	28%	85%
77%	97%	77%	96%	54%	88%	56%	92%
90%	97%	90%	95.5%	75%	79%	77%	87%
100%	96%	100%	94.5%	100%	83%	100%	84%
QMIX: StarCraft II							
Random		Timed		Counterfactual		Zero-sum	
44%	98%	51%	99%	28%	91%	39%	91%
65%	96.5%	73%	99%	49%	96%	63%	96%
94.5%	99%	99%	95%	68%	93%	72%	92%
100%	98%	100%	99%	100%	92%	100%	90%

5.1.2 Qualitative Performance and Behavioral Analysis. We evaluate the qualitative performance of c-MARL algorithms under our attacks using environment-specific metrics. In cooperative navigation environment, we measure the average number of the occupied landmarks by the cooperative agents. In physical deception environment, we measure the average distance between the cooperative agents and the target landmark. In StartCraft II, we use the team win rate to show the attack impact. Table 1 shows the performance of each attack. The number of occupied landmarks per timestep decreases as the compromised agent deviates more from its optimal policy. Similarly, the distance from the closest cooperative agent to the target landmark increases as we attack at more timesteps. In the StarCraft II, under the optimal QMIX policy, the c-MARL team of agents wins the battle around 95% of the time. As shown in the last part of Table 1, attacking during just 25% of timesteps can reduce the win rate to ,as low as, 18%, for the strategically timed attack in a white-box setting, and 13% in the black box setting. In all attack strategies, a 100% attack rate decreases the win rate to 0%, and in most attacks, a 75% attack rate also does that. Overall, under the compromised agent attacks, the win rate decreases as attack rate increases, and even a small attack rate has significant impact on the QMIX robustness.

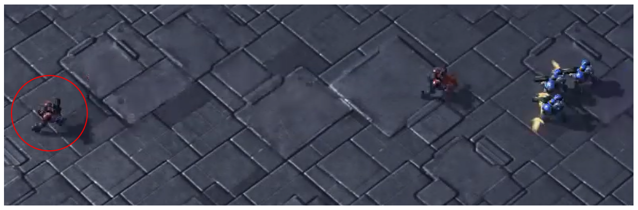
Qualitatively, in the cooperative navigation during the strategically-timed attack, the compromised agent learns to move away from the landmarks to distract its teammates from covering all landmarks. In the counterfactual reasoning based attack, the compromised agent displays interesting behaviors. It moves away moving from its landmark just before one of its teammate gets to a landmark which makes the affected agent confused about which landmark to cover and stood in between the 2 empty landmarks for the rest of the episode. This behavior suggests that the compromised agent succeeds by manipulating its teammate's observations through its actions. In this environment, there is not much room for the agents to recover from the compromised agent's adversarial actions. In the physical deception environment, we notice that when the other cooperative agent goes towards the non-target landmark, expecting the compromised agent to cover the target, it does not alter its behavior to cover the target. This is a robustness issue in MADDPG algorithm. Ideally, the agents should be robust enough to not have to rely on a teammate who is not doing its job. Similarly, we see that when the compromised agent leaves the non-target landmark uncovered and moves towards the target landmark, the adversary does not always take advantage of this clue. The optimal behavior of the adversary would be to move towards the only landmark which is being covered by a cooperative agent, instead it stays still. This behavior once again shows a lack of robustness in the agents under our attack.[1] In the StarCraftII environment, during all attack except the counterfactual reasoning based attack, we notice that the compromised agent moves away from its team until its teammates gets defeated. Then, it moves towards the enemies to get killed. In strategically-timed attack specifically, the compromised agent runs away only when its team is getting closer to the enemy (Figure 4a). In the counterfactual reasoning based attack, we notice sometimes the compromised agent luring one of its teammates to start attacking the enemy team then it itself runs away (Figure 4b) causing its team to get defeated.

5.2 Evaluation of The Evasion Technique

In this section, we show the detection results of our attacks using the developed c-MARL anomalous behavior detection model in Eq.4 (Sec.3.2). Then, we extend our attacks to become APTs by using the evasion technique (Sec.4.4).

In Table 2, we show the precision and recall of the attacks detection using our detection model (Eq.4) at different attacking rates. We use the performance metrics of recall and precision. Recall is the number of correctly identified anomalies in proportion to the number of true anomalies in the datasets. Precision is the proportion of correctly identified anomalies in relation to the number of predicted anomalies. Lower precision corresponds to higher false positive rates. The sensitive nature of these attacks makes recall paramount in performance metric. The detection model achieves better precision and recall when the attack rate increases across all attacks suggesting that at least 90% of the attacks can easily

[1]See Appendix C for screenshots of the compromised agent's behavior in the cooperative navigation and physical deception environments. The full demo videos are included in the supplementary materials


(a)

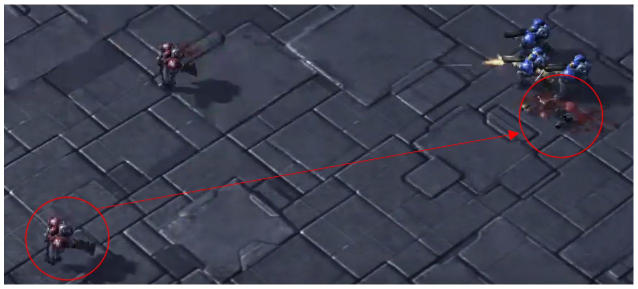
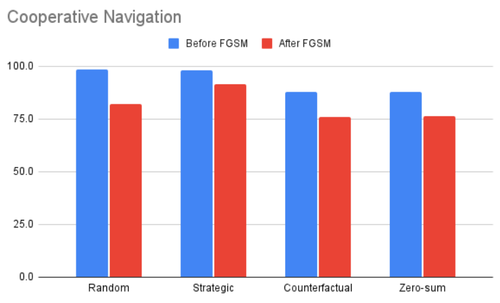

(b)

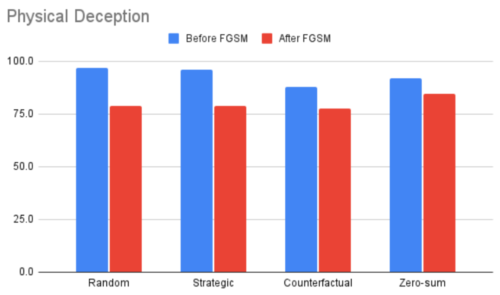
Figure 4: Screenshots from replayed episodes of StarCraft II.The compromised agent is controlled by an adversarial policy trained with (a) strategically-timed attack, and (b) counterfactual reasoning based attack

be traced and mitigated at the attack rate as low as of 25% of the episode time. Furthermore, the results show that the counterfactual reasoning based attack then the Zero-sum attack evade the detection more than other attacks with higher false negatives (i.e. smaller recalls) in all environments in the white and black box settings. This finding suggests that the self-destruction adversarial policies (i.e. randomly-timed and strategically-timed attacks) against c-MARL are easily detected. We believe that is because the adversarial policy for the compromised agent strongly deviates from its optimal policy to generate impactful harm to the system. On the other hand, the second category of the attacks (i.e. counterfactual reasoning based and Zero-sum attacks) that are designed to sabotage other agents' policies are more stealthy since the compromised agent deviates from its normal behavior only when its actions would negatively impact other agents in the system. Consequently, this deviation is not strong enough to get detected.

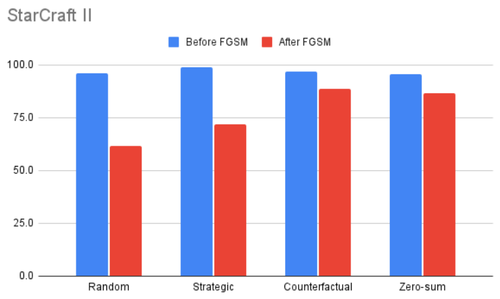
To increase the evasion of our attacks against the detection model, we enhance our attacks with the proposed evasion method in Sec.4.4. We found the most effective ϵ for the cooperative navigation, physical deception, and Starcraft II environments, are .35, .9, .7 (Figure 4 in Appendix A.2), respectively. Then, the compromised agent used those ϵ value to perturb its own states before passing them to the detection model. The only perturbed observations belonged to the compromised agent, assuming it can control the observation it reports to the detection model. Keeping all else constant, the evasion technique severely reduced the detection model's performance in detecting our attacks in all environments as shown in Figure 5. Timely-random attack achieved the higher evasion rate and we believe that is because the state perturbations adds more randomness to the whole system increases the detection complexity for the detection model.



(a)



(b)



(c)

Figure 5: The detection results of the c-MARL attacks in all environments before and after applying the evasion technique (Sec4.4) in the white-box settings. The results for the black-box are in Appendix A.2.

6 CONCLUSION

We investigate the maleficence of c-MARL by targeting and weaponizing one agent, termed *compromised*, to follow an adversarial policy that pushes activation of cooperating agents' policy networks off distribution. Using 2 particle environments and StarCraft II in a white-box and black-box settings, we proposed 4 attack strategies to control the compromised agent: 1) randomly-timed attack, 2) strategically timed attack, 3) counterfactual reasoning based attack, and 4)Zero-sum attack. We also extended those attacks with an evasion technique to create stealthy APTs. Those APTs evaded a strong detection model by at least 16% and at most 36%. Our attacks can be utilized to evaluate the robustness of c-MARL algorithms.

REFERENCES

[1] Pavel Filonov, Andrey Lavrentyev, and Artem Vorontsov. 2016. Multivariate Industrial Time Series with Cyber-Attack Simulation: Fault Detection Using an LSTM-based Predictive Data Model. arXiv:1612.06676 [cs.LG]

[2] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2017. Counterfactual Multi-Agent Policy Gradients. arXiv:1705.08926 [cs.AI]

[3] Jakob N. Foerster, Richard Y. Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. 2018. Learning with Opponent-Learning Awareness. arXiv:1709.04326 [cs.AI]

[4] Justin Fu, Katie Luo, and Sergey Levine. 2018. Learning Robust Rewards with Adversarial Inverse Reinforcement Learning. arXiv:1710.11248 [cs.LG]

[5] Adam Gleave, Michael Dennis, Cody Wild, Neel Kant, Sergey Levine, and Stuart Russell. 2020. Adversarial Policies: Attacking Deep Reinforcement Learning. arXiv:1905.10615 [cs.LG]

[6] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. arXiv:1412.6572 [stat.ML]

[7] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. 2017. Adversarial Attacks on Neural Network Policies. arXiv:1702.02284 [cs.LG]

[8] Uk Jo, Taehyun Jo, Wanjun Kim, Iljoo Yoon, Dongseok Lee, and Seungho Lee. 2019. Cooperative Multi-Agent Reinforcement Learning Framework for Scalping Trading. arXiv:1904.00441 [cs.AI]

[9] Jernej Kos and Dawn Song. 2017. Delving into adversarial attacks on deep policies. arXiv:1705.06452 [stat.ML]

[10] Sergey Levine. 2018. Reinforcement Learning and Control as Probabilistic Inference: Tutorial and Review. arXiv:1805.00909 [cs.LG]

[11] Jieyu Lin, Kristina Dzeparoska, Sai Qian Zhang, Alberto Leon-Garcia, and Nicolas Papernot. 2020. On the Robustness of Cooperative Multi-Agent Reinforcement Learning. arXiv:2003.03722 [cs.LG]

[12] Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. 2019. Tactics of Adversarial Attack on Deep Reinforcement Learning Agents. arXiv:1703.06748 [cs.LG]

[13] Minghuan Liu, Ming Zhou, Weinan Zhang, Yuzheng Zhuang, Jun Wang, Wulong Liu, and Yong Yu. 2020. Multi-Agent Interactions Modeling with Correlated Policies. arXiv:2001.03415 [cs.MA]

[14] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2020. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. arXiv:1706.02275 [cs.LG]

[15] G. Mclachlan. 1999. Mahalanobis Distance. *Resonance* 4 (06 1999), 20–26. https://doi.org/10.1007/BF02834632

[16] A. Peake, J. McCalmon, B. Raiford, T. Liu, and S. Alqahtani. 2020. Multi-Agent Reinforcement Learning for Cooperative Adaptive Cruise Control. In *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*. 15–22. https://doi.org/10.1109/ICTAI50040.2020.00013

[17] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. arXiv:1803.11485 [cs.LG]

[18] L. S. Shapley. 1953. Stochastic Games. *Proceedings of the National Academy of Sciences* 39, 10 (1953), 1095–1100. https://doi.org/10.1073/pnas.39.10.1095 arXiv:https://www.pnas.org/content/39/10/1095.full.pdf

[19] Yichuan Charlie Tang. 2020. Towards Learning Multi-agent Negotiations via Self-Play. arXiv:2001.10208 [cs.RO]

[20] Ermo Wei, Drew Wicke, David Freelan, and Sean Luke. 2018. Multiagent Soft Q-Learning. arXiv:1804.09817 [cs.AI]

[21] Ying Wen, Yaodong Yang, Rui Luo, Jun Wang, and Wei Pan. 2019. Probabilistic Recursive Reasoning for Multi-Agent Reinforcement Learning. arXiv:1901.09207 [cs.LG]

[22] Rey Wiyatno and Anqi Xu. 2018. Maximal Jacobian-based Saliency Map Attack. arXiv:1808.07945 [cs.LG]

[23] Yingce Xia, Tao Qin, Wei Chen, Jiang Bian, Nenghai Yu, and Tie-Yan Liu. 2017. Dual Supervised Learning. arXiv:1707.00415 [cs.LG]

[24] Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. 2018. Mean Field Multi-Agent Reinforcement Learning. arXiv:1802.05438 [cs.MA]

[25] Yiren Zhao, Ilia Shumailov, Han Cui, Xitong Gao, Robert Mullins, and Ross Anderson. 2019. Blackbox Attacks on Reinforcement Learning Agents Using Approximated Temporal Information. arXiv:1909.02918 [cs.LG]