Weaponizing Actions in Multi-Agent Reinforcement Learning: Theoretical and Empirical Study on Security and Robustness

Tongtong Liu[1], Joe McCalmon[1], Md Asifur Rahman[1], Cameron Lischke[2], Talal Halabi[3], and Sarra Alqahtani[1]

[1] Computer Science Department, Wake Forest University, Winston-Salem, NC, USA
liut18, mccalmonjoe,rahmm21,alqahtas@wfu.edu
[2] Computer Science Department, John Hopkins University, Baltimore, MD, USA
[3] Department of Computer Science and Software Engineering, Laval University, QC, Canada
talal.halabi@ift.ulaval.ca

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. This paper investigates the robustness of c-MARL to a novel adversarial threat, where we target and weaponize one agent, termed the *compromised agent*, to create natural observations that are adversarial for its team. The goal is to lure the compromised agent to follow an adversarial policy that pushes activations of its cooperative agents' policy networks off distribution. This paper shows mathematically the exploitation steps of such an 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, namely MADDPG and QMIX, to the compromised agent threat by deploying four attack strategies in three environments in white and black box settings. By targeting a single agent, our attacks yield highly negative impact on the overall team reward in all environments, reducing it by at least 33% and at most 89.6%. Finally, we provide recommendations on improving the robustness of c-MARL.

Keywords: Multi-Agent Reinforcement Learning · adversarial policies · robustness · security threats · compromised agent.

1 Introduction

The advances in single-agent Reinforcement Learning (RL) algorithms have sparked a new interest in cooperative Multi-Agent Reinforcement Learning (c-MARL). Several c-MARL training algorithms have been recently developed for deployment in several application domains such as cyber-physical systems [3], sensor networks, and social sciences. Nonetheless, these algorithms present new security vulnerabilities. For instance, it may not be surprising that adversarial attacks exist in RL to decrease the agent performance [8, 14]. However, the

vulnerability of the compromised agent in c-MARL systems has been only investigated once in a white-box setting against the QMIX algorithm [13] using a modified version of the adversarial example attack JSMA [24] in one environment. Therefore, it is important and necessary to develop a systematic methodology for designing non-trivial adversarial attacks which can efficiently and effectively exploit the vulnerabilities of c-MARL in order to design more robust algorithms.

Adversarial attacks on c-MARL are different from those carried out against individual RL agents [25, 8, 11]. First, each agent in c-MARL interacts with and modifies the state of the environment not only for itself but also for its cooperative agents. For an episode of L timesteps, an adversary has $2^L \times N$ choices of attacking or not attacking at least one agent at each timestep in a system having N agents, which significantly amplifies the attack surface for c-MARL compared to RL. Second, an adversary to c-MARL may have 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 the *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 is relevant to the robustness of multi-agent domains such as autonomous driving platoons [17], negotiation [21], and automated scalping trading [10]. 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; it can only modify the victim agent's observations via its own actions. For example, in autonomous vehicle platoons, pedestrians and other drivers cannot add noise to arbitrary pixels, or make a building disappear. Nonetheless, 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 the observations of a third party's orders.

In this paper, we present an algorithmic framework to study the robustness of c-MARL algorithms. First, we reverse engineer the c-MARL algorithms in the centralized-learning and decentralized-execution paradigm to exploit the compromised agent vulnerability. In this paradigm, the c-MARL algorithms use a centralised learning and make use of decentralised policies with only local observations during execution. Then, we empirically analyze the impact of this vulnerability on the robustness of two of the state-of-the-art algorithms in three different environments. We develop four attack strategies to exploit the compromised agent vulnerability in c-MARL in white-box and black-box settings, where the adversary has access to the reward function, state transition function, and the policies of all agents in the former settings and has no knowledge of these in the latter. Our contributions are threefold:

1. We show the feasibility of the compromised agent vulnerability by reverse-engineering the algorithms in the centralized-learning decentralized-execution paradigm of c-MARL.

2. We design four attack strategies to demonstrate the impact of the compromised agent vulnerability on the robustness of c-MARL algorithms.
3. We thoroughly analyze the robustness of the c-MARL algorithms MADDPG [16] and QMIX [18] under our attacks in three different environments involving cooperative and competitive tasks [4]. The designed attacks bring down the team reward in all environments by at least 33% and at most 89.6%.

The remainder of this paper is organized as follows. We discuss the background and related work on c-MARL robustness in Section 2. Section 3 formulates the threat model and introduces the proposed approach. Section 4 presents our experiments and discusses the results. We provide our recommendations in Section 5. Final remarks and conclusions are outlined in Section 6.

2 Background and Related Work

This section presents the background information necessary to fully understand our proposed attack strategies on c-MARL and discusses the related work.

2.1 c-MARL

In this paper, we model the c-MARL system using stochastic games [19]. For an n-agent stochastic game, we define a tuple:

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

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 where 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 [15]. At each stage of the game, actions are taken simultaneously. Each agent is assumed to pursue the maximal cumulative reward [20] expressed by:

$$\max \eta^i(\pi_{\theta^i}) = E\bigg[\sum_{t=1}^{\infty} \gamma^t r^i(s_t, a^i_t, a^{-i}_t)\bigg] \tag{3}$$

[4] https://github.com/SarraAlqahtani22/MARL-Robustness

with (a_t^i, a_t^{-i}) sample from $(\pi_{\theta^i}^i, \pi_{\theta^{-i}}^{-i})$. Correspondingly, for a game with an infinite time horizon, the state-action Q-function can be defined by $Q_{\pi_\theta^i}^i(s_t, a_t^i, a_t^{-i}) = E\Big[\sum_{l=0}^{\infty} \gamma^l r^i(s_{t+1}, a_{t+1}^i, a_{t+1}^{-i})\Big]$.

The centralized-learning and decentralized-execution paradigm of MARL has been followed by the major c-MARL algorithms including MADDPG [16], COMA [5], MF-AC [27], and Q-Mix [18]. Although those algorithms simplify the learning in multi-agent environments by using the non-correlated factorization of the joint policy, they are vulnerable to the compromised agent attack shown in this paper. This vulnerability arises when the implicit connections among the agents' actions are ignored. Agents trained with such algorithms cannot efficiently identify the normal and abnormal behaviors of other agents working together to accomplish certain tasks. This paper focuses on attacking MADDPG [16] and QMIX [18], the lead algorithms in the centralized-learning and decentralized-execution paradigm.

2.2 Related Work on Adversarial Attacks in MARL

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 [8]. Other attacks reduce the number of adversarial examples needed to decrease the agent's reward [14] or trigger its misbehavior [28]. However, the literature did not study the security of c-MARL and the effects of cooperation on the success of attacks. The closest work to our proposed threat model and attack strategies proposes a two-step attack against QMIX with the objective of reducing the total team reward by perturbing the observation of a single agent [13]. The authors extend an existing adversarial example method, namely JSMA [24], 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.

The approach proposed in this paper is fundamentally different since it involves 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 three different environments. Also, compared to the work of Lin et al. [13] which focuses on attacks in the competitive multi-agent setting, our threat model considers cooperative teams of agents in both fully-cooperative and competitive environments.

3 Proposed Adversarial Approach

We first describe our threat model then mathematically explain how an adversary can exploit the compromised agent vulnerability and carry out various attacks.

3.1 Threat Model

We assume that the attacker can take over at least one benign agent (the compromised agent m) and use it to create naturally adversarial observations via its actions to attack the other agents $-m$ in the system. Recall that the joint transition probability in c-MARL is $p(s_{t+1}|s_t, a^1, ..., a^n)$, indicating that the attack surface for c-MARL includes any perturbations to the state s_t through adversarial examples and any perturbations to the other agents' actions $a^1, ..., a^n$ through the compromised agent attacks. This threat targets the agents' actions instead of the states' features usually targeted by adversarial examples [8, 14, 28]. The attacker can only intercept the compromised agent's actions via network attacks such as hijacking, impersonation, and Man in The Middle (MiTM). Studies have shown that multi-agent systems are vulnerable to such attacks [4, 1, 7]. The attacker's goal is to weaponize actions taken by the compromised agent to distract team members from accomplishing their task either by preventing them from converging to the optimal policy or misleading them to a dangerous state.

In the white-box setting, the adversary may have access to the environment's reward function, the policies of every agent in the system, and the ground truth state transition probability $p(s_{t+1}|s_t, a_0, \ldots, a_n)$, where s_t is the concatenation of each agents' observations $(o_1, \ldots, o_n)$. Conversely, a black box adversary has no access to the internal configuration, rewards, or policies of benign agents including the compromised one. We also assume that 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 [25]. Since the agents' policies are held fixed, the Markov game G in Eq. (1) reduces to a single-player MDP, denoted by $G_m = (S, A_m, T_m, R_m)$, 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 $-m$.

The compromised agent problem in c-MARL is defined by:

$$\max \sum_{t=0}^{t=T} \mathbf{KL}\ (p(a_t^{-m}|a_t^m, s_t)||p(a_t^{-m}|a_t^{*m}, s_t)) \tag{4}$$

where a^{*m} represents the adversarial actions generated by the adversarial policies for the compromised agent m. This equation 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 agents $-m$, $p(a_{t+1}^{-m}|a_t^{*m}, s_t^m)$, pushing the activations of their policy networks off distribution. Practically, we solve the problem in Eq. (4) by finding the adversarial actions a^{*m} for the compromised agent following one of the proposed attack strategies in Section 3.3 to maximize the KL-divergence.

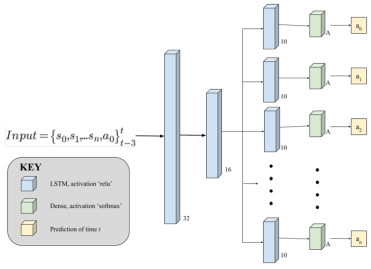

Fig. 1. The architecture of the deep learning-based behavioral cloning.

3.2 Exploiting The 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 engineer the learning algorithm by correlating the learning process of agent m's policy with the learning of the conditional policies of agents $-m$. First, we collect state-action pairs by observing the 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 for training an Adversarial Inverse RL (AIRL) algorithm [6] to discover the hidden reward function behind agent m's behavior, $r^m(s_t, a_t^m, a_t^{-m}, s_{t+1})$, while considering the actions taken by the 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 can be reformulated as [23, 26]:

$$\begin{aligned}\pi_\theta(a^m, a^{-m}|s) &= \underbrace{\pi^m_{\theta^m}(a^m|s)\pi^{-m}_{\theta^{-m}}(a^{-m}|s, a^m)}_{\text{Compromised agent's perspective}} \\ &= \underbrace{\pi^{-m}_{\theta^{-m}}(a^{-m}|s)\pi^m_{\theta^m}(a^m|s, a^{-m})}_{\text{other agents' perspective}}\end{aligned} \tag{5}$$

From the perspective of the compromised agent m, the first equality in Eq. (5) indicates that the joint policy can be essentially decomposed into two parts: agent m's 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 environment state and agent m's action, i.e., what the compromised agent believes the other agents might think based on their original policy.

In the white-box setting, 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 of the transition function and the conditional policies of the victims $\pi^{-m}_{\theta^{-m}}(a^{-m}|s, a^m)$ via the behavioral cloning model depicted in Fig. 1, which 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 different learnable parameters Φ:

$$T_\Phi = p(s_{t+1}|s_t, a^1, ..., a^n) \tag{6}$$

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

$$\arg\max_{\theta^m, \phi^{-m}} \left(\pi^m_{\theta^m}(a^m|s) \rho^{-m}_{\phi^{-m}}(a^{-m}|s, a^m) \right) \tag{7}$$

With the learning protocol in Eq. (7), 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 probabilistic RL [12]. We first derive the probability of τ being observed during the reconnaissance phase as follows:

$$p(\tau) = \left[p(s_1) \prod_{t=1}^{T} p(s_{t+1}|s_t, a^m_t, a^{-m}_t) \right] \exp\left(\sum_{t=1}^{T} r^m(s_t, a^m_t, a^{-m}_t) \right) \tag{8}$$

Since we used AIRL to discover r^m using τ, the goal becomes to find 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. (7) such that the induced trajectory distribution $\hat{p}(\tau)$ can match the ground-truth of trajectory probability $p(\tau)$:

$$\hat{p}(\tau) = p(s_1) \prod_{t=1}^{T} p(s_{t+1}|s_t, a^m_t, a^{-m}_t) \pi^m_{\theta^m}(a^m_t|s_t) \rho^{-m}_{\phi^{-m}}(a^{-m}_t|s_t, a^m_t) \tag{9}$$

We can now optimize the approximated policies of the victim agents by minimizing the KL-divergence between Eq. (8) and Eq. (9):

$$\begin{aligned} KL(\hat{p}(\tau)||p(\tau)) = \\ -E_{\tau\ \hat{p}(\tau)}[log p(\tau) - log \hat{p}(\tau)] = -\sum_{t=1}^{t=T} E_{\tau\ \hat{p}(\tau)} \left[r^m(s_t, a^m_t, a^{-m}_t) + \right. \\ \left. H\left(\pi^m_{\theta^m}(a^m_t|s_t) \rho^{-m}_{\phi^{-m}}(a^{-m}_t|s_t, a^m_t) \right) \right] \end{aligned} \tag{10}$$

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. (10) yields the optimal Q-function for agent m (Theorem 1 in [23]):

$$Q^m_{\pi_\theta} = log \int_{a^{-m}} E(Q^m_{\pi_\theta}(s, a^m, a^{-m})) da^{-m} \tag{11}$$

The corresponding $-m$ conditional policy becomes:

$$\rho^{-m}_{\phi^{-m}}(a^{-m}_t|s_t, a^m_t) = \frac{1}{Z} E(Q^m_{\pi_\theta}(s, a^m, a^{-m}) - Q^m_{\pi_\theta}(s, a^m)) \tag{12}$$

To solve Eq. (12), we maintain two Q-functions and iteratively update them using the learned reward function for agent m and the observations in τ as the

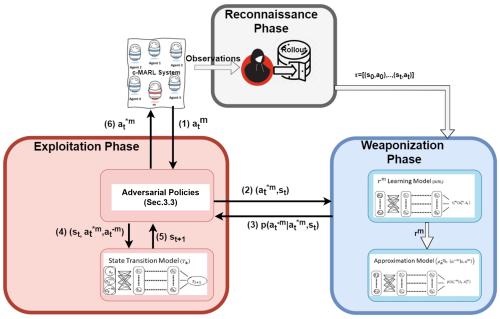

Fig. 2. The steps of exploiting the compromised agent vulnerability.

ground-truth. Eq. (11) approximates the original policy of m while considering the approximated conditional policies of agents $-m$ in Eq. (12). The overall steps of exploiting the compromised agent vulnerability are shown in Fig. 2. For the proof of Eq. (11) and Eq. (12), please check [23].

3.3 Attack Strategies

According to Fig. 2 and Eq. (10), we still need to generate the adversarial policies for the compromised agent to choose a_t^{*m}. In this section, we introduce our attack strategies to generate those policies under two different categories: attacks based on the compromised agent's self-destruction strategies, and attacks based on destructive strategies of other agents' objectives.

Self-Destruction Adversarial Policies This category focuses on minimizing the compromised agent's individual reward which indirectly reduces the reward of the c-MARL team.

Randomly-Timed Attack: We attack the victims' policies by developing a set of randomized off-distribution adversarial policies for the compromised agent to sabotage the c-MARL system through its own actions. At a certain percentage of timesteps, the adversary changes the compromised agent's action into a_t^{*m} based on a random off-distribution policy.

Strategically-Timed Attack: This attack strategically selects a subset of timesteps to change the compromised agent's actions. We first calculate the c-function [14]:

$$c(s_t) = \max_{a_t}(\pi_m(s_t, a_t)) - \min_{a_t}(\pi_m(s_t, a_t)) \tag{13}$$

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; otherwise, its accumulated reward will be reduced.

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's 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 of the system. We design a reward function for an RL agent to generate a long-term policy that replaces the original actions $a_t^m, ..., a_{t+l}^m$ to the best set of adversarial actions $a_t^{*m}, .., a_{t+l}^{*m}$. The RL agent would choose certain timesteps to attack instead of attacking each timestep based on how much divergence the attack is expected to produce. To do so, the reward function is crafted as follows:

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

where a^{*m} represents the set of counterfactual actions to the action a^m generated by the original policy of agent m, $\pi_{\theta^m}^m$, and γ is a discount factor. The attack starts at timestep t, and l is the number of steps into the future when the attacker wants 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. We formulate this attack as a single agent RL problem to minimize the cumulative reward for the whole team as:

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

where a_t^{*m} is the compromised agent's action at timestep t, θ^m parameterizes the compromised agent's policy, and r_t is the global reward recovered by [6]. (a_t^{*m}, a_t^{-m}) is sampled from $(\pi_{\theta^m}^m, \pi_{\theta^{-m}}^{-m})$ in the white box setting. In the black box setting, we sample from the approximated policies derived in Eq. (11) and (12). 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 Experiments and Results

This section describes the implementation of our attacks and discusses the obtained results. [5]

[5] Our code and demos are available here:https://github.com/SarraAlqahtani22/MARL-Robustness

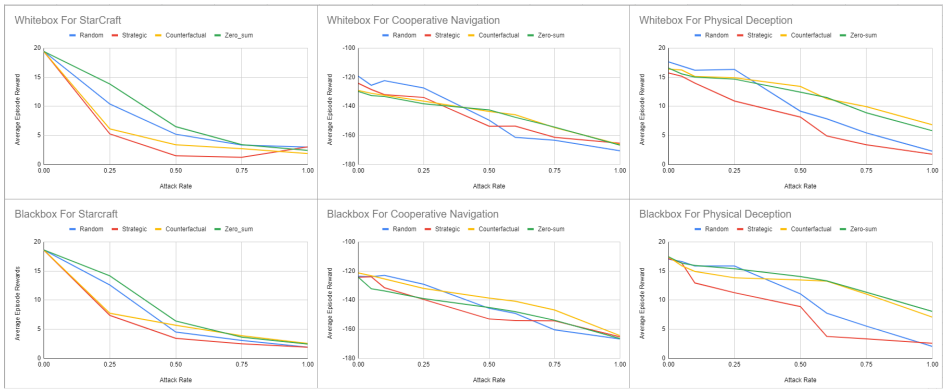

Fig. 3. The average episode rewards in 2 particle multi-agent environments (MADDPG) and StarCraft2 (QMIX) under our attacks as a function of attack rate.

4.1 Experimental Setup

We evaluate the robustness of the MADDPG algorithm under our attacks in both the white and black box settings in two particle environments [16]: cooperative navigation and physical deception with variant attacking rates. In the cooperative navigation environment, 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 $N = L = 2$.

We evaluate the robustness of QMIX using StarCraft II, 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. 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.

4.2 Results and Discussion

We discuss the obtained results from two perspectives: the adversarial policies and c-MARL team reward, and the qualitative performance and behavioral analysis of the agents.

Table 1. The average number of occupied landmarks in the cooperative navigation, average distance between the cooperative agents and target landmark in physical deception [16], and win rate for the cooperative team in StarCraft II [18] for 25%, 50%, 75%, and 100% attack rates across all attack types in 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

Adversarial Policies and Team Reward To learn an adversarial policy for the compromised agent, we used the attack strategies explained in Section 3.3. 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 at 0% and moving to 100% by increments of 25%. We present the team average reward for the victim agents in Fig. 3. As the attack rate increases, the team reward decreases, indicating that each attack was successful at degrading the robustness of c-MARL algorithms. The strategically-timed attack has the highest negative impact on the 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, the strategically-timed attack achieved 86.6% and 85% reward drop for 100% attack rate in white-box and black-box settings, respectively. In the cooperative navigation environment, the 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 as opposed to the randomly-timed attack which attacks at random timesteps during the episode.

The counterfactual reasoning-based attack is able to achieve similar levels of performance with respect to the 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 rates.

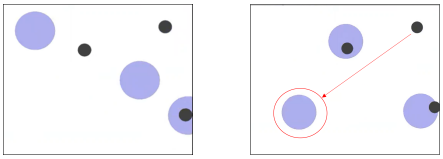

Fig. 4. Screenshots from replayed episodes of the cooperative navigation environment from MADDPG under the strategically-timed attack.

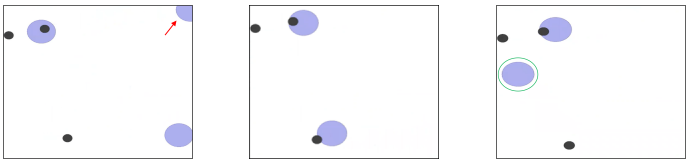

Fig. 5. Screenshots from replayed episodes of the cooperative navigation environment from MADDPG under the counterfactual reasoning-based attack.

We believe since those environments are purely cooperative, the adversarial policies trained to directly sabotage other agents' policies can better diminish the robustness of the algorithms in cooperative settings. However, in the physical deception environment that naturally involves an enemy, the counterfactual and zero-sum attacks have the least impact on the team reward.

Although the results in Fig. 3 and Table 1 show that the counterfactual reasoning-based attack succeeds, we observe an effect similar in nature to the canonical imitation learning problem. As we start attacking the system using the compromised agent's counterfactual actions, the impact of the attack was not as strong as the other attacks. We hypothesize that this is due to the inability of the behavioral cloning model to accurately capture the other agents' states that were not part of their optimal policies during reconnaissance. In general, the canonical problem of imitation learning accumulates the learning errors, resulting in the learner encountering unknown states [2]. Moreover, this attack builds its adversarial policy based on predicting each agent's action for a sequence of timesteps then finds the counterfactual actions with maximum KL divergence using those predictions. Hence, the prediction errors accumulate over the timesteps and hinder the attack.

Qualitative Performance and Behavioral Analysis We evaluate the qualitative performance of c-MARL algorithms under our attacks using environment-specific metrics. In the cooperative navigation environment, we measure the average number of the occupied landmarks by the cooperative agents. In the 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 evaluate the attack impact. Table 1 shows the performance of each attack. In cooperative navigation, the number of occupied landmarks per timestep decreases as the compromised agent deviates more from its optimal policy. Similarly, in physical deception, the distance from the closest cooperative agent to

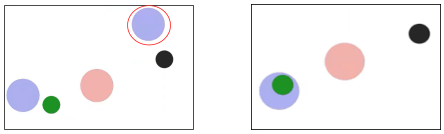

Fig. 6. Screenshots from replayed episodes of the physical deception environment from MADDPG under the strategically-timed attack. To the left, we see the beginning of the episode where the cooperative agents are splitting over the target and non-target landmarks. To the right, the compromised agent is moving away from the non-target landmark, leaking the information to the adversary.

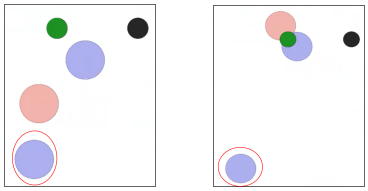

Fig. 7. Screenshots from replayed episodes of the physical deception environment from MADDPG under the counterfactual reasoning-based attack. To the left, we see the beginning of the episode. To the right, the compromised agent is moving away from the non-target landmark at a perfect timing to leak the information to the adversary which got the clue and moved towards the target.

the target landmark increases as we attack at more timesteps.

In 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 by at least 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 the attack rate increases, and even a small attack rate has significant impact on the QMIX robustness.

We show here the qualitative analysis of the cooperative navigation environment under the strategically-timed and counterfactual reasoning-based attacks, chosen to represent the attacks in each category. During the strategically-timed attack, the compromised agent immediately moves away from the landmarks to prevent its teammates from covering all landmarks. In this environment, there is not much room for the agents to recover from the compromised agent's adversarial actions. The left figure in Fig. 4 shows the beginning of the episode where the agents attempt to cover the 3 landmarks (black circles). Then, in the right figure, the compromised agent (within the red circle) is moving away from its supposedly assigned landmark.

In the counterfactual reasoning-based attack, the compromised agent displays interesting behaviors. The first behavior is shown in Fig. 5. The compromised agent (pointed to with a red arrow) is moving away from the landmark just before its teammate(s) get to a landmark, which makes the affected agent (within

Fig. 8. Screenshots from replayed episodes of StarCraft II. The compromised agent is controlled by an adversarial policy trained with the strategically-timed attack (top) and counterfactual reasoning-based attack (bottom).

a green circle) confused about which landmark to cover (stands in between the 2 empty landmarks for the remaining of the episode). This behavior suggests that the compromised agent succeeds by manipulating its teammate's observations through its actions. We notice another behavior under this attack strategy where the compromised agent moves to the same landmark that has been already covered by one of its teammate causing the team reward to decrease.

In the physical deception environment, during the strategically-timed attack, we notice that when the compromised agent moves away from the non-target landmark (Fig. 6), the other cooperative agent does not alter its behavior to move away from the target. This is a robustness issue in the MADDPG algorithm. Ideally, the agents should be robust enough to not have to rely on a teammate who is not doing its job. Similarly, 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 (Fig. 7). The optimal behavior of the adversary would be to move towards the only landmark which is being covered by a cooperative agent, instead it stands still. This behavior once again shows a lack of robustness in the agents under our attack.

In the StarCraftII, during all attacks except the counterfactual reasoning-based attack, we notice that the compromised agent moves away from its team until its teammates get defeated. Then, it moves towards the enemies to get killed. In the strategically-timed attack, the compromised agent runs away only when its team is getting closer to the enemy (top of Fig. 8). In the counterfactual reasoning-based attack, we notice that sometimes the compromised agent is luring one of its teammates to start attacking the enemy team then it itself runs away (bottom of Fig. 8) causing its team to get defeated.

5 Recommendations

The idea that slightly perturbing one compromised agent's observations can reduce performance at such a large scale is alarming. With the seemingly heavy

dependence on small fractions of the global observations, attackers can implement similar attacks to go virtually unnoticed by detection models. By only compromising one agent, the entire system will be affected. And without reliable detection, it is clear that these attacks can be devastating.

The naive remedy once a compromised agent has been detected is to exclude it from the team and mark it as an adversary. However, the deviation in one agent's behavior could occur due to a security threat or its lack of knowledge about the environment which is reasonable considering the limited vision range and communication bandwidth in MARL systems. Thus, we think it is critical to extend the assured RL (ARL) techniques into c-MARL by embedding real-time formal verification methods in c-MARL algorithms, which would shield the agents from reaching the dangerous states by training them to trade off between performance and security/safety in the environments containing either malicious (adversarial) or faulty (dysfunctional) agents. Those methods should be developed for run-time verification focusing only on the agent's time-bounded, short-term behavior for scalability. This will increase the robustness of MARL systems functioning in uncertain and unpredictable environments.

To solve the non-correlated factorization problem of the MARL centralized learning-decentralized execution paradigm, the other agents' actions were taken into consideration by decoupling the joint policy as a correlated policy conditioned on the environment state and other agents' actions [23, 9, 22]. Those algorithms assume that agents participating in a joint activity act rationally and cooperate to achieve shared goals. However, relying only on this assumption to achieve joint goals without the establishment and reinforcement of trust opens a vulnerability similar to the one discussed here. Hence, studying the compromised agent vulnerability in this paradigm of c-MARL algorithms becomes essential.

6 Conclusion

We investigate the maleficence of c-MARL by targeting and weaponizing one agent, termed *compromised*, to follow an adversarial policy that pushes the activation of the cooperating agents' policy networks off distribution. We proposed four 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. Our attacks reduce the overall team reward in all environments by at least 33% and at most 89.6%. In future work, we intend to develop more robust c-MARL algorithms by accounting for the compromised agent vulnerability during training. This will increase the robustness of c-MARL systems functioning in uncertain environments. Moreover, our attacks can be utilized to evaluate the robustness of c-MARL algorithms. We will make our code and demos available in the final submission.

References

1. Amoozadeh, M., Raghuramu, A., Chuah, C., Ghosal, D., Zhang, H.M., Rowe, J., Levitt, K.: Security vulnerabilities of connected vehicle streams and their impact

on cooperative driving. IEEE Communications Magazine **53**(6), 126–132 (2015). https://doi.org/10.1109/MCOM.2015.7120028
2. Bagnell, J.A.D.: An invitation to imitation. Tech. Rep. CMU-RI-TR-15-08, Carnegie Mellon University, Pittsburgh, PA (March 2015)
3. Bakakeu, J., Kisskalt, D., Franke, J., Baer, S., Klos, H.H., Peschke, J.: Multi-agent reinforcement learning for the energy optimization of cyber-physical production systems. In: 2020 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE). pp. 1–8 (2020). https://doi.org/10.1109/CCECE47787.2020.9255795
4. Dadras, S., Dadras, S., Winstead, C.: Collaborative attacks on autonomous vehicle platooning. In: 2018 IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS). pp. 464–467 (2018). https://doi.org/10.1109/MWSCAS.2018.8624026
5. Foerster, J.N., Farquhar, G., Afouras, T., Nardelli, N., Whiteson, S.: Counterfactual multi-agent policy gradients. AAAI (2018)
6. Fu, J., Luo, K., Levine, S.: Learning robust rewards with adversarial inverse reinforcement learning (2018)
7. Higgins, F., Tomlinson, A., Martin, K.M.: Survey on security challenges for swarm robotics. In: 2009 Fifth International Conference on Autonomic and Autonomous Systems. pp. 307–312 (2009). https://doi.org/10.1109/ICAS.2009.62
8. Huang, S., Papernot, N., Goodfellow, I., Duan, Y., Abbeel, P.: Adversarial attacks on neural network policies (2017)
9. Jaques, N., Lazaridou, A., Hughes, E., Gulcehre, C., Ortega, P.A., Strouse, D., Leibo, J.Z., de Freitas, N.: Social influence as intrinsic motivation for multi-agent deep reinforcement learning (2019)
10. Jo, U., Jo, T., Kim, W., Yoon, I., Lee, D., Lee, S.: Cooperative multi-agent reinforcement learning framework for scalping trading (2019)
11. Kos, J., Song, D.: Delving into adversarial attacks on deep policies (2017)
12. Levine, S.: Reinforcement learning and control as probabilistic inference: Tutorial and review (2018)
13. Lin, J., Dzeparoska, K., Zhang, S.Q., Leon-Garcia, A., Papernot, N.: On the robustness of cooperative multi-agent reinforcement learning (2020)
14. Lin, Y.C., Hong, Z.W., Liao, Y.H., Shih, M.L., Liu, M.Y., Sun, M.: Tactics of adversarial attack on deep reinforcement learning agents (2019)
15. Liu, M., Zhou, M., Zhang, W., Zhuang, Y., Wang, J., Liu, W., Yu, Y.: Multi-agent interactions modeling with correlated policies (2020)
16. Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., Mordatch, I.: Multi-agent actor-critic for mixed cooperative-competitive environments. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. p. 6382–6393. NIPS'17, Curran Associates Inc., Red Hook, NY, USA (2017)
17. Peake, A., McCalmon, J., Raiford, B., Liu, T., Alqahtani, S.: Multi-agent reinforcement learning for cooperative adaptive cruise control. In: 2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI). pp. 15–22 (2020). https://doi.org/10.1109/ICTAI50040.2020.00013
18. Rashid, T., Samvelyan, M., de Witt, C.S., Farquhar, G., Foerster, J., Whiteson, S.: Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning (2018)
19. Shapley, L.S.: Stochastic games. Proceedings of the National Academy of Sciences **39**(10), 1095–1100 (1953). https://doi.org/10.1073/pnas.39.10.1095, https://www.pnas.org/content/39/10/1095

20. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. A Bradford Book, Cambridge, MA, USA (2018)
21. Tang, Y.C.: Towards learning multi-agent negotiations via self-play (2020)
22. Tian, Z., Zou, S., Davies, I., Warr, T., Wu, L., Ammar, H.B., Wang, J.: Learning to communicate implicitly by actions (2019)
23. Wen, Y., Yang, Y., Luo, R., Wang, J., Pan, W.: Probabilistic recursive reasoning for multi-agent reinforcement learning (2019)
24. Wiyatno, R., Xu, A.: Maximal jacobian-based saliency map attack (2018)
25. Wu, X., Guo, W., Wei, H., Xing, X.: Adversarial policy training against deep reinforcement learning. In: 30th USENIX Security Symposium (USENIX Security 21). pp. 1883–1900. USENIX Association (Aug 2021), https://www.usenix.org/conference/usenixsecurity21/presentation/wu-xian
26. Xia, Y., Qin, T., Chen, W., Bian, J., Yu, N., Liu, T.Y.: Dual supervised learning (2017)
27. Yang, Y., Luo, R., Li, M., Zhou, M., Zhang, W., Wang, J.: Mean field multi-agent reinforcement learning (2018)
28. Zhao, Y., Shumailov, I., Cui, H., Gao, X., Mullins, R., Anderson, R.: Blackbox attacks on reinforcement learning agents using approximated temporal information (2019)