THE SECURITY AND ROBUSTNESS OF MULTI-AGENT REINFORCEMENT LEARNING (MARL) ALGORITHMS

BY

TONGTONG (FRANK) LIU

A Thesis Submitted to the Faculty of the

DEPARTMENT OF COMPUTER SCIENCE AT WAKE FOREST UNIVERSITY

in Partial Fulfillment of the Requirements for

Graduation with Honors in

Computer Science

May 2023

Winston-Salem, North Carolina

Approved By:

Sarra Alqahtani, Ph.D., Thesis Advisor

Grey Ballard, Ph.D., Academic Advisor

Daniel Cañas, Ph.D., Chair of Honors Committee

William Turkett, Ph.D., Department Chair

Acknowledgments

I would like to express my deepest gratitude to my research advisor, Dr. Sarra Alqahtani, who has been an unwavering source of support throughout my academic journey. It has been my honor to be one of her first students in the lab. Dr. Alqahtani's dedication to research and her students is unparalleled and has been a constant source of motivation for me. Without her help, I would not be the person I am today, and I would not have achieved so many milestones.

I also had the pleasure of working with many great students in the group. Special thanks to Joe McCalmon and Asifur Rahman, who helped me in various aspects of research and acted as role models for me to learn and improve.

I would also like to thank the Department of Computer Science at Wake Forest University, where I learned to write my first line of code and eventually graduated with the John W. Sawyer Prize as the top senior CS graduate. The education, resources, and opportunities provided to me by the department were truly life-changing.

This endeavor would not have been possible without the support of my parents, Mrs.Yali Yang and Mr.Yuetao Liu, and my girlfriend, Emily Shang, who provided me with endless love and encouragement outside of academic work. They supported me in all of my pursuits and decisions, and their belief in me has been my anchor throughout this journey.

Contents

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 security robustness of issues 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 of *compromised agent* is to lure the compromised agent to follow an adversarial policy that pushes activations of its cooperative agents' policy networks off distribution. This paper also propose CMDP Multi-Agent Particles environment by adding safety constraints in original particle environments [1] and train agents in CMDP Multi-Agent Particles environments to proof the convergence property so it can be used as a baseline for future researchers. We theoretically form the ideas of *safety certificates* so that we train agents in a two-step framework. With safety certificates, agents learn to behavior in a safer way by assuming the worst of others by first learning unsafe behavior and then learn to act differently from unsafe behaviors. This paper shows mathematically the exploitation steps of compromised agent 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.

1 Introduction

The advances in single-agent Reinforcement Learning (RL) algorithms have sparked a new interest in cooperative Multi-Agent Reinforcement Learning(c-MARL) systems. c-MARL systems contain a team of intelligent agents working together towards a shared goal, and each agent's state is part of other agent's observation, where all the agents are trained to maximize the sum of their accumulated rewards. c-MARL training algorithms are widely applicable to cyber-physical systems, sensor networks, and social sciences. One of the most interesting applications is autonomous driving car. However, security and robustness need to be examined before any wide range deployment of such system into reality, or this may cause inestimable damage to the society.

At the same time, the prevalence of Deep Reinforcement Learning (DRL) brought significant technological advancements in the AI community and different state-of-the-art algorithms allow more effective training and more promising results. However, a common perturbation in deep learning model, adversarial examples (carefully crafted noise that is invisible to human eye) [2], has been shown to be effective in attack DRL system as well. Specifically, a DRL agent's policy performance significantly decreased by only introducing relatively small perturbations in the inputs to the DNN, either consistently [3] or strategically [4]. However, their work mainly focused on single agent system but did not systematically study the security of c-MARL and the effects of cooperation on

the success of attacks. The closest work to c-MARL system only reduces the total team reward by perturbing the observation of a single agent [5]. Those works were on the theoretical side of the attack, and they assumed attackers knew all the information, for example the team-reward function and the environment transition function (white-box setting), which is uncommon in real practices. Therefore, to design a physically realistic attack model that does not depend on directly modifying the agents' observation with adversarial examples is important in MARL setting.

This work achieve this by only weaponizing one agent to follow a trained adversarial policy that confuses its teammate by pushing activations of its cooperative agents' policy network off distribution. The primary motivation of this work is to develop a systematic methodology for designing non-trivial adversarial attacks which can efficiently and effectively exploit the vulnerabilities of c-MARL to design more robust algorithms. Such method is termed *compromised agent vulnerability*. This idea is related to domains such as autonomous driving platoons [6] and automated scalping trading [7] where 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 financial trading, an adversary can send orders to an exchange market which will appear in the victim's market data feed, but the attacker cannot modify the observations of a third party's orders.

Furthermore, after successfully attacking the system, defense method needs to be develop to potentially mitigate attacks. This work proposed the idea of *safety certificate* to ensure safety against sensor noise, adversarial examples, and malicious actions by compromised agent during runtime. The objective for the certificate is to develop a lower bound of safety by assuming other agents in c-MARL system perform the worst-case actions. With safety certificate, agents are assumed to be *cautious* about their teammate's action until trust can be established through observations of a series of actions. The idea of safety certificate is inherent in human's cognitive ability. For example, assumes a world that autonomous driving car is widely deployed, and the car is trained to follow traffic rules. When a pedestrian want to cross the street, the pedestrian will never risk his or her life by crossing the street without ensuring the car stop in front of the zebra crossing. In human's cognitive consciousness, we are always cautious about our surroundings and assume the worst of others until trust can be established. Therefore, safety certifications uses similar ideas to make sure agents act 'cautiously' and think 'vigilantly' to its teammate's actions. This thesis answered two questions: 1) How can we define the *worst* behavior of others? and 2) How can we be 'cautious' to those behaviors?

To summarize, this work will primarily contains two part: 1) we designed a systematic attack methodology that exploit the vulnerability of c-MARL system for MADDPG [1] and QMix [8] Algorithm and empirically analyze the impact of this vulnerability in three different RL environments. We designed four specific attack strategies to exploit the compromised agent vulnerability in c-MARL in both white-box and black-box settings, and 2) we designed a defense mechanism in c-MARL system for proposed attack called safety certificate. We proposed the three-step theoretical framework for safety certificate that extends the idea of AdvEx-RL [9] from single-agent to

multi-agent setting, modified the multi-agent particle environments into Constraint Markov Decision Process (CMDP) setting, and prove the convergence property of training c-MARL agents in CMDP RL environments. We show that training agents until optimal policy doesn't necessarily means safety is also ensured. The contributions are summarized as followed:

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 [1] and QMIX [8] under our attacks in three different environments involving cooperative and competitive tasks. The designed attacks bring down the team reward in all environments by at least 33% and at most 89.6%.
4. We proposed the Multi-Agent CMDP environments baseline by modifying the multi-agent particle environments and prove the convergence properties.
5. We proposed the three step theoretical framework for adding safety certificate to agents in c-MARL system.

2 Background and Related Works

This section presents the background information necessary to fully understand our proposed attack strategies on c-MARL and CMDP, and discusses the related work on security and robustness in c-MARL system.

2.1 Reinforcement Learning (RL)

RL provides an agent with a framework to interact with its surrounding environment and receive feedback signals. RL algorithms use these feedback signals to improve the decision-making policy of the agent. More specifically, the agent interacts with the environment in a series of discrete timesteps, indexed by $t = 0, 1, 2, 3, \ldots, T$, where T is the end of a single environment trajectory called an episode. At each timestep within an episode, the agent observes a representation of the environment called the state and denoted by $s_t \in S$, where S is the set of all possible states. The agent then chooses an action $a_t \in A$, where A is the set of all possible actions, using its policy π. The probability that action a will be chosen by the agent given state s_t is denoted by $\pi(a|s_t)$. The environment's transition function decides the resulting next state in timestep $t + 1$, where $p(s_{t+1}|a, S_t)$ is the probability of state s_{t+1} resulting from an agent taking action a in state s_t. Following each action, the agent receives a reward based on the outcome of its action in s_t, hence $R_{t+1} = R(s_t, s_t)$. The goal of an RL agent is to maximize the expected sum of rewards over the

course of an episode, or the return denoted by $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$.

2.2 Cooperative Multi-Agent Reinforcement Learning (c-MARL)

In this paper, we model the c-MARL system using stochastic games [10]. 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 $T : S \times A_1 \times A_2 .. \times A_n \to \triangle(S)$ 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 [11]. At each stage of the game, actions are taken simultaneously. Each agent is assumed to pursue the maximal cumulative reward [12] expressed by:

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

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

The centralized-learning and decentralized-execution paradigm of MARL has been followed by the major c-MARL algorithms including MADDPG [1], COMA [13], MF-AC [14], and Q-Mix [8]. 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 [1] and QMIX [8], the lead algorithms in the centralized-learning and decentralized-execution paradigm.

2.3 Constraint Markov Decision Process (CMDP)

We consider the standard CMDPs, $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mu, \mathcal{P}(.|.,.), \mathcal{R}, \gamma, \mathcal{C})$ where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action space; μ and $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ denote the initial state distribution and state transition dynamics respectively. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{R}$ is the reward function; γ denotes discount factor; and $\mathcal{C} = \{c_i : \mathcal{S} \times \mathcal{A} \xrightarrow{\mathcal{S}} R \geq 0; i = 1, 2, .., T\}$ denotes the set of cost associated with safety constraint violations in any trajectory episode $\tau = \{s_0, a_0, ..., a_{T-1}, s_T\}$ with a maximum trajectory length of T. We assume that accomplishing the task goal or violating a safety constraint in $\mathcal{M}$ leads to episode termination.

2.4 Related Work on Adversarial Attacks and Defense 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 [3]. Other attacks reduce the number of adversarial examples needed to decrease the agent's reward [4] or trigger its misbehavior [15]. 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 [16]. The authors extend an existing adversarial example method, namely JSMA [17], 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. [16] 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.

Previous defense work has also been focused on single-agent setting. The idea of Adversarial Behavior Exclusion framework (AdvEx-RL) trains an adversarial policy to interactively extract unsafe behaviors followed by learning a safety policy by maximizing the divergence from such adversarial policy [9]. The approach in this paper proposed a great framework of ensuring safety against unsafe behaviors. However it is only focused on single-agent setting that considers unsafe behaviors of itself. Another paper develop an online certifiable robust for DRL algorithms that consider to create a observation perturbations to defense against noise or adversarial examples [18]. However, directly modifying an agent's observation (state) is unrealistic in real-world setting, as mentioned in Section 1. These two works motivate us to develop a action perturbation method in MARL settings against malicious actions taken by one of the agents using counterfactual reasoning about the worst actions that can be taken by other agents.

3 Method: Compromised Agent Vulnerability

We first describe the threat model and assumptions we made during attack and defense. We will then mathematically explain how an adversary can exploit the compromised agent vulnerability.

3.1 Threat Model

We have theoretically formed the threat model for this attack. 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. This threat targets the agents' actions instead of the states' features usually targeted by adversarial examples [2] [3] [4]. 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 [19]. 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 function $p(s_t + 1|s_t, a_0, \ldots, a_n)$, 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 pre-trained 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 [20].

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.

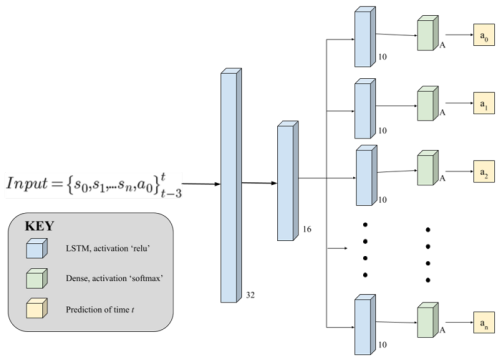

Figure 1: The architecture of the deep learning-based behavioral cloning.

(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.

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 [21] 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 [22, 23]:

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

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 $\pi_{\theta^{-m}}^{-m}(a^{-m}|s, a^m)$ 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 $\rho_{\phi^{-m}}^{-m}(a^{-m}|s, a^m)$. 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:

$$\arg\max_{\theta^m, \phi^{-m}} \left(\pi_{\theta^m}^m(a^m|s) \rho_{\phi^{-m}}^{-m}(a^{-m}|s, a^m) \right) \tag{7}$$

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 [24]. We first derive the probability of τ being observed during the reconnaissance phase as follows:

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

Since we used AIRL to discover r^m using τ, the goal becomes to find the best approximation of $\pi_{\theta^m}^m(a_t^m|s_t)\rho_{\phi^{-m}}^{-m}(a_t^{-m}|s_t, a_t^m)$ to maximize Eq. (7) such that the induced trajectory distribution $\hat{p}(\tau)$ can match the ground-truth of trajectory probability $p(\tau)$:

$$\hat{p}(\tau) = p(s_1) \prod_{t=1}^T p(s_{t+1}|s_t, a_t^m, a_t^{-m}) \pi_{\theta^m}^m(a_t^m|s_t) \rho_{\phi^{-m}}^{-m}(a_t^{-m}|s_t, a_t^m) \tag{9}$$

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

$$\begin{aligned} KL(\hat{p}(\tau)||p(\tau)) = \\ -E_{\tau\ \hat{p}(\tau)}[log p(\tau) - log \hat{p}(\tau)] = -\sum_{t=1}^{t=T} E_{\tau\ \hat{p}(\tau)} \big[r^m(s_t, a_t^m, a_t^{-m}) + \\ H\left(\pi_{\theta^m}^m(a_t^m|s_t) \rho_{\phi^{-m}}^{-m}(a_t^{-m}|s_t, a_t^m) \right) \end{aligned} \tag{10}$$

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 [22]):

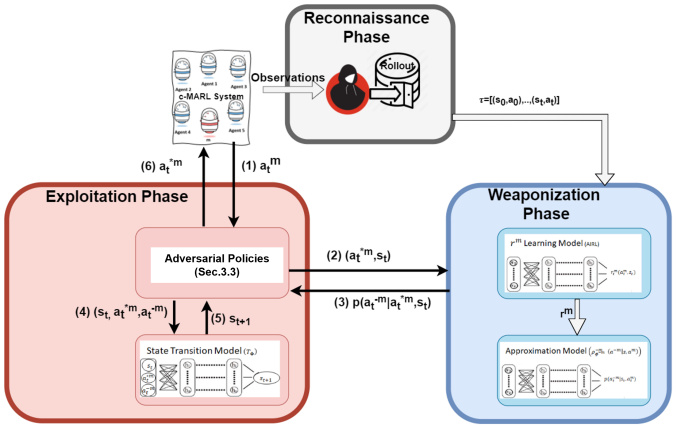

Figure 2: The steps of exploiting the compromised agent vulnerability.

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

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 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 [22].

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^{*m}_t. 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.

3.3.1 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 [4]:

$$\max_{a_t}(\pi_m(s_t, a_t)) - \min_{a_t}(\pi_m(s_t, a_t)) > \beta \tag{13}$$

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.

3.3.2 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'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 [21]. (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 Method: Safety Certificates

4.1 CMDP for Multi-Agent Particle Environments

Constraint Markov Decision Process (CMDP) are environments that agents interacts with under certain constraints. It sets a specific goal for the agents to learn (through reward function) while associated cost with any safety violation behavior. For example, in continuous MuJoCo CMDPs (Maze, Navigation 1, and Navigation 2), agents need to reach the goal while also avoiding collisions with obstacles in the environment [25]. Based on the best of our knowledge, however, there are no CMDP environments in multi-agent settings. We constructed standard CMDPs for nine Multi-Agent Particle Environment [1] by adding safety violations to the environments (details to be find in the original paper). The environments, some of which shown on Figure 3, typically consists of N agents and L landmarks in a two-dimensional world with continuous space and discrete time. Some environments are fully cooperative (all agents need to maximize a shared return) or cooperative-competitive (some agents have conflicting goals). As an example, the third image of Figure 3, Cooperative Navigation, is a fully cooperative environments where agents must cooperative through taking actions to over all the landmarks. They need to observe the positions of others learns to 'cover' all the landmarks.

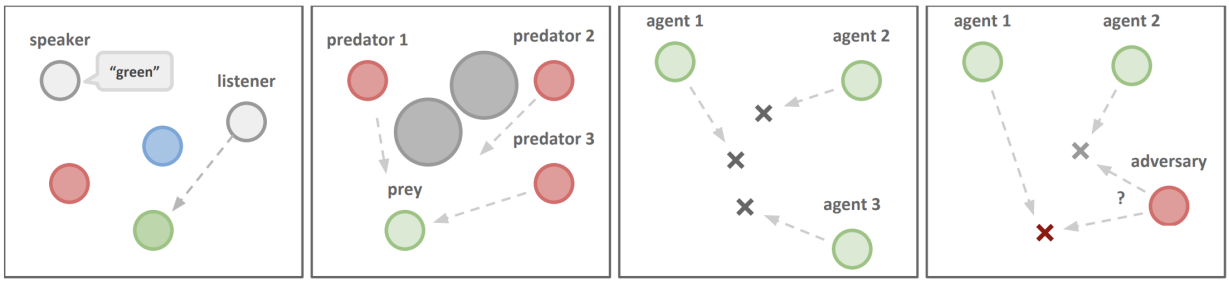

Figure 3: Visualization from left to right: 1) Cooperative Communication (simple speaker listener), 2) Predator-Prey (simple tag), 3) Cooperative Navigation (simple spread), and d) Physical Deception (simple adversary).

Assume N agents are in the environments, cost c is defined for each individual agents, and can be reported individually for cooperative-competitive environments, and both individually or as a sum

for fully cooperative environments. The value of c can be adjusted by researchers considering how heavy they want to penalize unsafe behavior, and it is defined to be 5 in this paper. The value of c decides the balance point of the joint optimization process described in Section 4.3. It is also worth to mention that the cost is a parse cost, not a dense cost, and it is a function similar to reward, which defined as $c(s_t, a_t)$. Based on the nature of those nine environments, we categorized them into three categories in terms of defining cost and reconfigure the environments into CMDPs. Table 1 summarize the categorization, and more detailed are explained as follows.

Environments	**Cost Categorization**
Simple adversary	Collision with agent
Simple spread	
Simple crypto	
Simple tag	Collision with obstacles
Simple push	
Simple world common	
Simple reference	Stay within the non-communication zone
Simple speaker listener	

Table 1: Summary of environments and the cost categorization

4.1.1 Collision with Agents

For environments that involves task of navigation (multiple agents need to navigate to multiple goals), collision with each other is an unsafe behavior that is not preferably to observe. In fact, simple adversary, simple spread, and simple crypto falls into this category. For example, simple spread (cooperative navigation) involves agents navigate to different targets. Collision, therefore, would be an unsafe behavior. A similar example in the real world is the multi-agent autonomous platoon [6]. An integer value c is associated with every collision happened in one episode.

4.1.2 Collision with Obstacles

For environments that are cooperative-competitive (involves both good agents and adversary in one game) that involves chasing one another, collision with each other is part of the game. Therefore the cost is defined to be colliding with randomly generated obstacles inside the map of the game. A similar example in real world is the predator-prey situation in wild world. The predator need to catch the prey while also avoiding colliding with trees and other obstacles in the forest. An integer value c is associated with every collision happened in one episode.

4.1.3 Non-communication Zones

For environments that involves communications between agents (through sending private code), cost is defined to be staying inside of randomly generated non-communication zones. Inside the

communication zone, the private code will be intercepted and replace to NULL. A similar example in real world would be when agent enters an area that have no internet access created by the attacker. An integer value c is associated with every timestep in one episode the agent stays inside the zone.

4.2 Learn to Violate Safety

In Multi-agent CMDP environments, one agent in the team, termed *unsafe agent* learns an optimal policy π^{adv*} to maximize the accumulated cost associated with the safety violation through the exploration-exploitation principle of RL. Before the unsafe behavior training, all other agents are trained using MADDPG until optimal with a policy π^* and remain fixed during the unsafe behavior training. Therefore, the training process is simplfied into a single-agent RL problem. Note if the environment is cooperative-competitive, the *unsafe agent* must be one of the good agent. In this process, the agent learns all possible ways to violate the safety constraint in the CMDP environments.

4.3 Learn to Be Cautious

The trained optimal unsafe behavior policy π^{adv*} will be used as a heuristic policy in this step to train the safe policy $\pi^{safe}(a_t|s_t)$ using a joint optimization process with two objectives.

Objective 1: Maximizing the KL Divergence From $\pi^{\mathbf{adv}*}$: KL Divergence of an arbitrary policy with respect to π^{adv*} guarantees to reduce the probability of selecting a trajectory leading to safety violations in single agent situation [9]. When other team member's policy is fixed, the problem becomes an single-agent problems so the assumptions holds. The first objective is to train $\pi^{safe}(a_t|s_t)$ in order to maximize:

$$\mathrm{D}_{\mathrm{KL}}(\pi^{safe}(a_t|s_t)\|\pi^{adv*}(a_t|s_t)) \tag{16}$$

However, maximizing the distance between two policies only guarantees a different a_t is generated from π^{safe} such that a_t is as different from unsafe action as possible, but such action is not proven to be safe. Then we need to also to optimize Objective 2 to ensure safety.

Objective 2: Minimize the Advantage Function $\mathbf{A_{adv}(s_t, a_t)}$

Advantage Function $A(s, a)$ is a value that reflect how many 'advantage' you gain through taking action a at state t, which mathematically defined as $A(s, a) = Q(s, a) - V(s)$. If we select action a_t that minimize $A_{adv}(s, a_t)$, the goal is to be 'disadvantaged' from the heuristic unsafe policy π^{adv*}. This minimization problem ensures minimum additional value is gained referenced upon unsafe behavior policy, therefore safety is ensured.

5 Experiments and Results

This section describes the implementation of our attacks and initial experiments with safety certificate, as well as discusses the obtained results.

5.1 Attack

5.1.1 Environment Setup

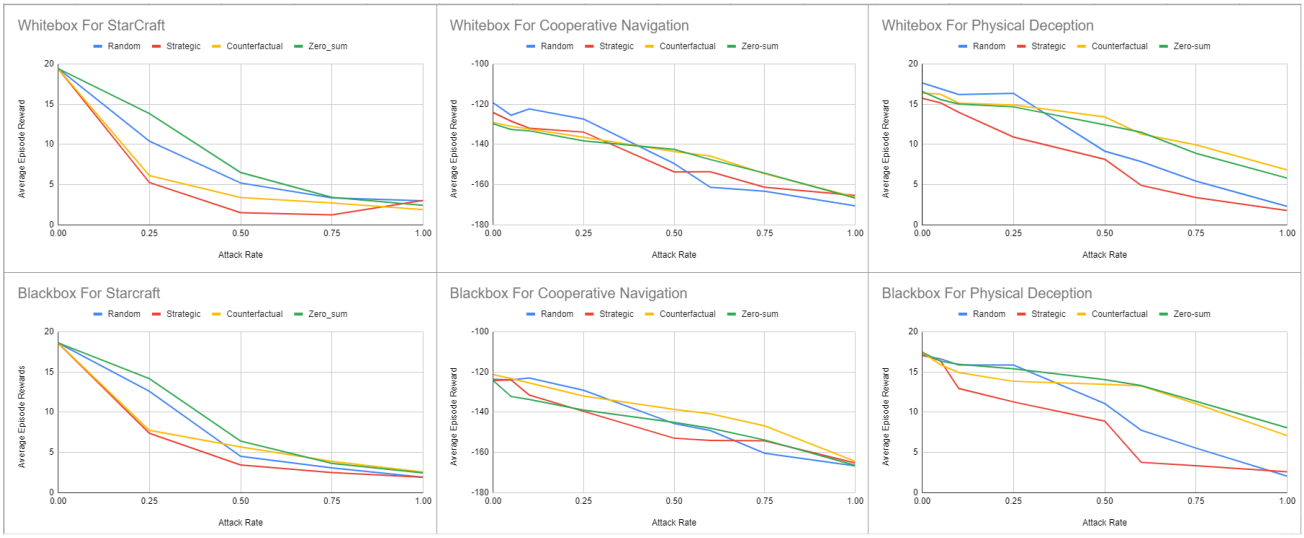

Figure 4: The average episode rewards in 2 particle multi-agent environments (MADDPG) and StarCraft2 (QMIX) under our attacks as a function of attack rate.

We evaluate the robustness of the MADDPG algorithm under our attacks in both the white and black box settings in two particle environments [1]: 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

Table 2: The average number of occupied landmarks in the cooperative navigation, average distance between the cooperative agents and target landmark in physical deception [1], and win rate for the cooperative team in StarCraft II [8] 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

marines, which use fixed policies. For our attacks, we control one of the cooperative marines, and alter its actions using the aforementioned attack strategies.

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.

5.1.2 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. 4. 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%

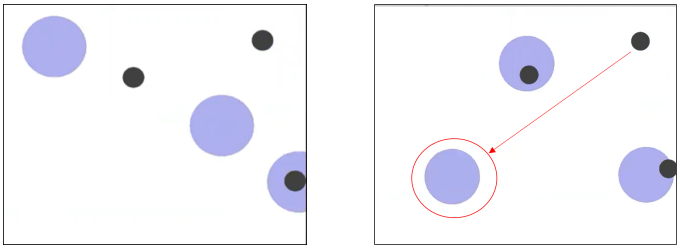

Figure 5: Screenshots from replayed episodes of the cooperative navigation environment from MADDPG under the strategically-timed attack.

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. 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. 4 and Table 2 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 [26]. 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.

5.1.3 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

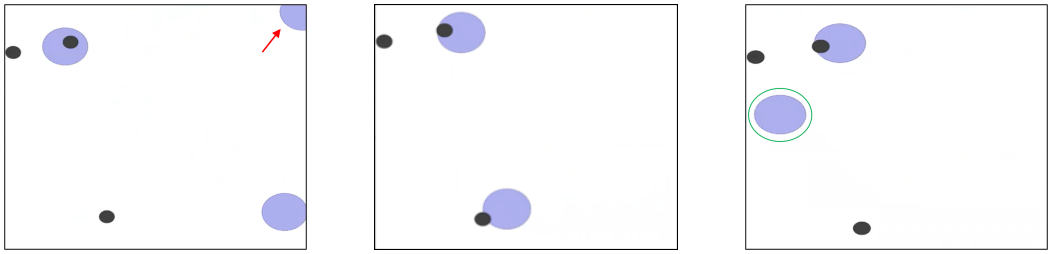

Figure 6: Screenshots from replayed episodes of the cooperative navigation environment from MADDPG under the counterfactual reasoning-based attack.

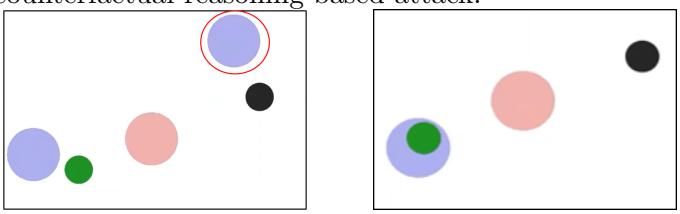

Figure 7: Screenshots from replayed episodes of the physical deception environment from MADDPG under the strategically-timed attack. To the left, we see the beginning of the episode where the cooperative agents are splitting over the target and non-target landmarks. To the right, the compromised agent is moving away from the non-target landmark, leaking the information to the adversary.

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 2 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 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 2, 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

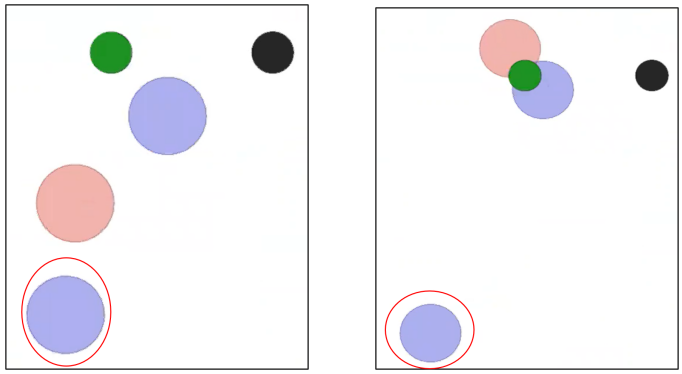

Figure 8: Screenshots from replayed episodes of the physical deception environment from MADDPG under the counterfactual reasoning-based attack. To the left, we see the beginning of the episode. To the right, the compromised agent is moving away from the non-target landmark at a perfect timing to leak the information to the adversary which got the clue and moved towards the target.

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. 5 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. 6. 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 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. 7), 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. 8). 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.

Figure 9: 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).

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. 9). 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. 9) causing its team to get defeated.

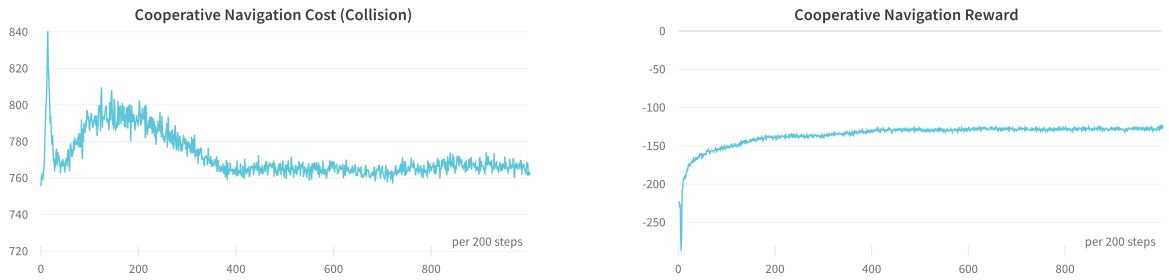

Figure 10: Cost and reward for good agents in simple spread during training

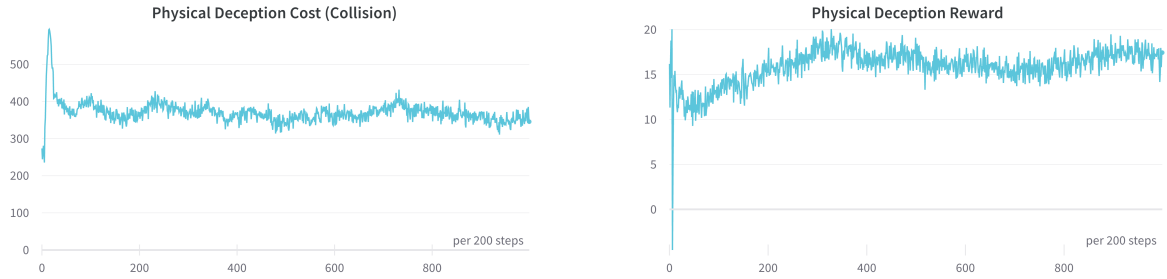

Figure 11: Cost and reward for good agents in simple adversary during training

5.2 Defense

5.2.1 CMDP Convergence Training

After modifying the Multi-Agent Particle environments [1] to become CMDP, convergence property has to be proved to ensure changes in environments does not violate the training assumptions by MADDPG. We followed the centralized training decentralized execution framework and procedures from [1] to train agents until convergence in eight different CMDP particle environments. We show the trend of good agent reward and good agent cost with respect to timestep.

Fig. 10 and Fig. 11 are the result from simple spread (cooperative navigation) and simple adversary (physical deception). We observed that the reward for the agents are trained to converge eventually, and the numbers matches the reported optimal reward from [1]. Therefore the modifications to Multi-Agent Particle environments into CMDP do not affect the convergence property of MADDPG training, and the agents are able to reach the same level of performance as before. Note that the value of cost doesn't become part of the reward function of the environments. In fact, the environment dynamic and reward function has remained unchanged. However, we observe a significant value of cost associated with the agents during training and even when they converges to optimal policy. This means the agents, even if are capable of acting optimally to achieve the goal, is not safe, because there are significant amount of cost reported during training. For example, in cooperative navigation, agents are able to cover landmarks as quickly as possible, but we still observe a significant amount of collisions, which prompt us to also consider if there are needs to sacrifice some of the performance to ensure safety.

At the same time , we also hope that as the first set of CMDP environments in Multi-agent domains, CMDP Multi-Agent Particle environments can be used as a baseline for future research on safety issue in MARL system.

Table that summarizes hyperparameters used during training and training figures for the rest of the environments can be found in the Appendix.

5.2.2 Future Work on Defense

Due to time limitations, the rest of the work will be taken by Wake Forest University, Computer Science master student Asifur Rahman. We have formulate the theoretical ideas and framework for safety certificate in Section 4. Eventually after implementing the idea of safety certificates, we hypothesized the optimal reward for each environment will be lower than the standard value (before safety certificates), but the cost will significantly dropped to close to zero. The central idea is to sacrifice some of the performance to ensure safety, which is a more realistic situation in the real life.

6 Conclusion

In this work, 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 propose 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%. We also proposed the idea of *safety certificates* to train the agent in a two-step process so that agents are capable of being 'catutious' to their teammate's behavior. We construct CMDP Multi-agent Particles environment as a baseline for future research. 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. The code is available in my Github: https://github.com/frank47ltt/MARL_Robustness.

References

[1] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *ArXiv*, vol. abs/1706.02275, 2017.

[2] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *CoRR*, vol. abs/1312.6199, 2014.

[3] S. H. Huang, N. Papernot, I. J. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," *ArXiv*, vol. abs/1702.02284, 2017.

[4] Y.-C. Lin, Z.-W. Hong, Y.-H. Liao, M.-L. Shih, M.-Y. Liu, and M. Sun, "Tactics of adversarial attack on deep reinforcement learning agents," in *IJCAI*, 2017.

[5] J. Lin, K. Dzeparoska, S. Q. Zhang, A. Leon-Garcia, and N. Papernot, "On the robustness of cooperative multi-agent reinforcement learning," *2020 IEEE Security and Privacy Workshops (SPW)*, May 2020. [Online]. Available: http://dx.doi.org/10.1109/SPW50608.2020.00027

[6] A. Peake, J. McCalmon, B. Raiford, T. Liu, and S. Alqahtani, "Multi-agent reinforcement learning for cooperative adaptive cruise control," in *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, 2020, pp. 15–22.

[7] U. Jo, T. Jo, W. Kim, I. Yoon, D. Lee, and S. Lee, "Cooperative multi-agent reinforcement learning framework for scalping trading," 2019.

[8] T. Rashid, M. Samvelyan, C. S. D. Witt, G. Farquhar, J. N. Foerster, and S. Whiteson, "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning," *ArXiv*, vol. abs/1803.11485, 2018.

[9] A. Rahman, T. Liu, and S. Alqahtani, "Adversarial behavior exclusion for safe reinforcement learning," in *Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI 2023)*, 2023.

[10] L. S. Shapley, "Stochastic games," *Proceedings of the National Academy of Sciences*, vol. 39, no. 10, pp. 1095–1100, 1953. [Online]. Available: https://www.pnas.org/content/39/10/1095

[11] M. Liu, M. Zhou, W. Zhang, Y. Zhuang, J. Wang, W. Liu, and Y. Yu, "Multi-agent interactions modeling with correlated policies," 2020.

[12] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[13] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," 2017.

[14] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean field multi-agent reinforcement learning," 2018.

[15] Y. Zhao, I. Shumailov, H. Cui, X. Gao, R. Mullins, and R. Anderson, "Blackbox attacks on reinforcement learning agents using approximated temporal information," 2019.

[16] J. Lin, K. Dzeparoska, S. Q. Zhang, A. Leon-Garcia, and N. Papernot, "On the robustness of cooperative multi-agent reinforcement learning," 2020.

[17] R. Wiyatno and A. Xu, "Maximal jacobian-based saliency map attack," 2018.

[18] M. Everett, B. Lütjens, and J. P. How, "Certifiable robustness to adversarial state uncertainty in deep reinforcement learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 9, pp. 4184–4198, 2022.

[19] M. Amoozadeh, A. Raghuramu, C.-N. Chuah, D. Ghosal, H. M. Zhang, J. Rowe, and K. N. Levitt, "Security vulnerabilities of connected vehicle streams and their impact on cooperative driving," *IEEE Communications Magazine*, vol. 53, pp. 126–132, 2015.

[20] A. Gleave, M. Dennis, N. Kant, C. Wild, S. Levine, and S. J. Russell, "Adversarial policies: Attacking deep reinforcement learning," *ArXiv*, vol. abs/1905.10615, 2020.

[21] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adversarial inverse reinforcement learning," 2018.

[22] Y. Wen, Y. Yang, R. Luo, J. Wang, and W. Pan, "Probabilistic recursive reasoning for multi-agent reinforcement learning," 2019.

[23] Y. Xia, T. Qin, W. Chen, J. Bian, N. Yu, and T.-Y. Liu, "Dual supervised learning," 2017.

[24] S. Levine, "Reinforcement learning and control as probabilistic inference: Tutorial and review," 2018.

[25] B. Thananjeyan, A. Balakrishna, S. Nair, M. Luo, K. Srinivasan, M. Hwang, J. E. Gonzalez, J. Ibarz, C. Finn, and K. Goldberg, "Recovery rl: Safe reinforcement learning with learned recovery zones," 2021.

[26] J. A. D. Bagnell, "An invitation to imitation," Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-15-08, March 2015.

7 Appendix

7.1 Statement of Published Work

This thesis incorporated two accepted publications from Tongtong Liu pertinent to the subject of the thesis, and Tongtong Liu had the major role in preparation of those manuscripts. Two pertinent papers are referenced as followed.

1. **Liu, T.**; McCalmon, J.; Lischke, C.; Rahman, A.; Halabi, T.; and Alqahtani, S. "Weaponizing Actions in Multi-Agent Reinforcement Learning: Theoretical and Empirical Study on Security and Robustness". Accpeted by The 24th International Conference on Principles and Practice of Multi-Agent Systems (PRIMA 2022).
2. Rahman, A.; **Liu, T.**; and Alqahtani, S. "Adversarial Behavior Exclusion for Safe Reinforcement Learning". In Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI 2023).

7.2 Additional Graph for CMDP Training

Training in CMDP Multi-Agent Particles environments fixed all the policies except for one good agent and using the following hyperparameters:

- maximum episode length: 25
- number of episode of training: 200000
- learning rate: 0.01
- gamma: 0.95
- batch size: 1024

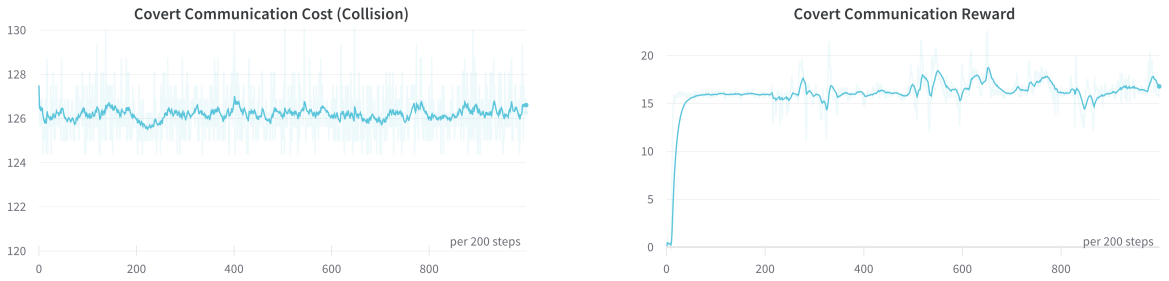

Figure 12: Cost and reward for good agents in simple crypto during training

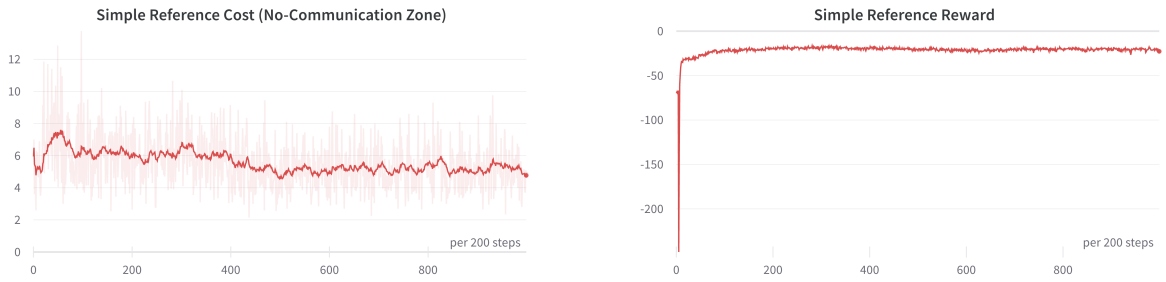

Figure 13: Cost and reward for good agents in simple reference during training

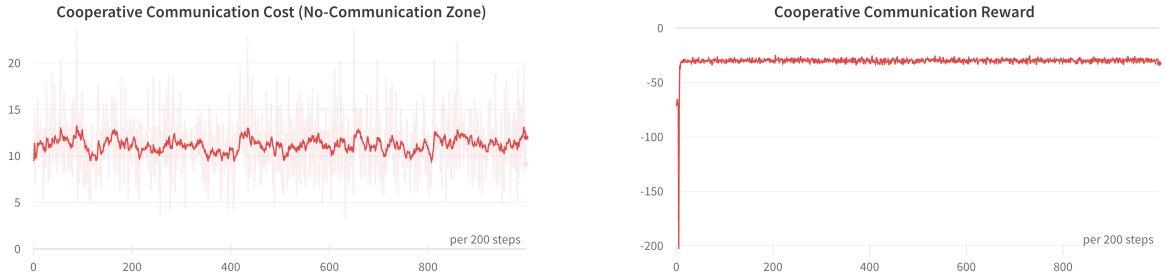

Figure 14: Cost and reward for good agents in simple speaker listener during training

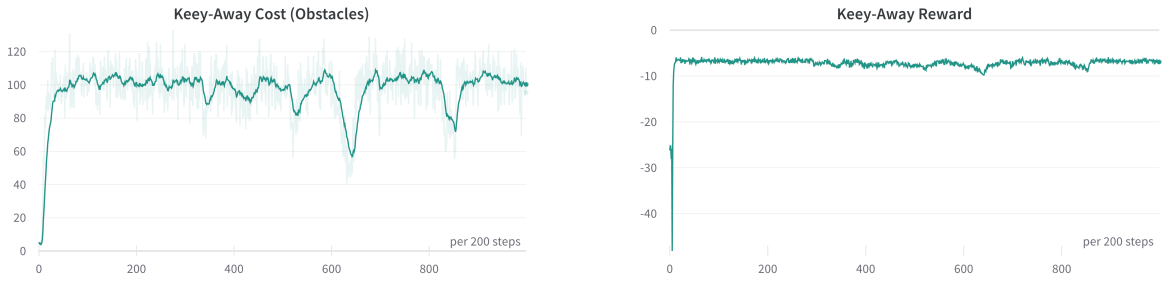

Figure 15: Cost and reward for good agents in simple push during training

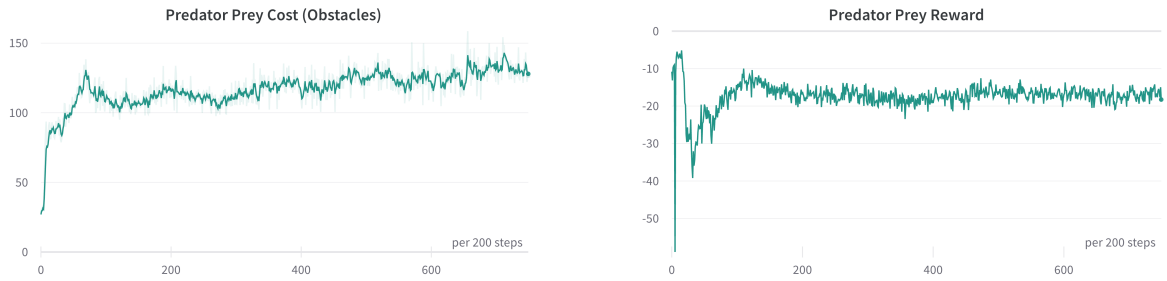

Figure 16: Cost and reward for good agents in simple tag during training

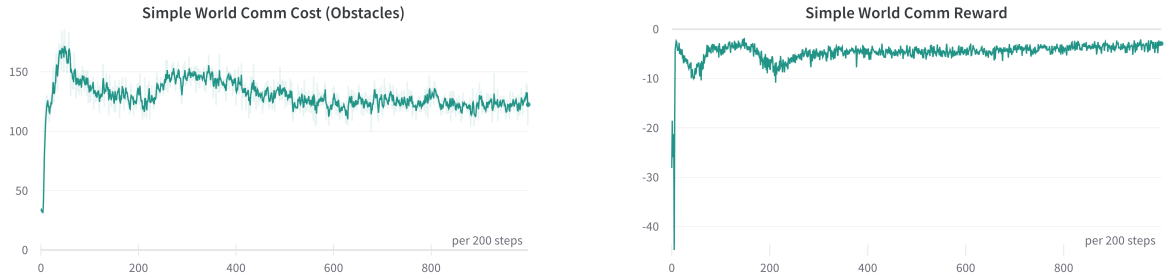

Figure 17: Cost and reward for good agents in simple world comm during training