Backdoor Attacks on Safe Reinforcement Learning-Enabled Cyber–Physical Systems

Shixiong Jiang, *Graduate Student Member, IEEE*, Mengyu Liu, *Graduate Student Member, IEEE*, and Fanxin Kong, *Member, IEEE*

***Abstract*— Safe reinforcement learning (RL) aims to derive a control policy that navigates a safety-critical system while avoiding unsafe explorations and adhering to safety constraints. While safe RL has been extensively studied, its vulnerabilities during the policy training have barely been explored in an adversarial setting. This article bridges this gap and investigates the training time vulnerability of formal language-guided safe RL. Such vulnerability allows a malicious adversary to inject backdoor behavior into the learned control policy. First, we formally define backdoor attacks for safe RL and divide them into active and passive ones depending on whether to manipulate the observation. Second, we propose two novel algorithms to synthesize the two kinds of attacks, respectively. Both algorithms generate backdoor behaviors that may go unnoticed after deployment but can be triggered when specific states are reached, leading to safety violations. Finally, we conduct both theoretical analysis and extensive experiments to show the effectiveness and stealthiness of our methods.**

***Index Terms*—Backdoor attack, cyber–physical systems, safe reinforcement learning.**

I. Introduction

CYBER–PHYSICAL systems (CPSs) integrate computing and networking components to control the physical system and interact with the environment using sensors and actuators. Researchers have been making efforts to embed artificial intelligence (AI) in CPS to enable applications such as autonomous vehicles, drones, and smart manufacturing [1]. However, the increasing autonomy also brings up new security and safety concerns for CPS [2], [3], [4].

Deep reinforcement learning (DRL) has demonstrated notable efficacy in resolving decision-making problems, specifically in acquiring control policies within simulated environments through iterative trial and error. Such success motivates the investigations into the deployment of DRL in real-world scenarios. However, conventional DRL has no safety considerations, and ensuring safety is important for real-world applications. Consequently, the concept of safe reinforcement learning (safe RL) has been introduced to derive a control policy that optimizes task performance and incorporates safety constraints during the training process.

There are two main research directions in safe RL. The first one solves the problem using a mathematical model describing how the system works [5], [6], [7]. The second one does not require such knowledge and instead follows a set of rules written in formal languages, e.g., linear temporal logic (LTL) [8] or signal temporal logic (STL) [9]. Safety requirements are formally specified and the specifications are used to guide the policy training.

Both directions leverage neural networks (NNs) as function approximations. However, DRL has been proven to be vulnerable to training time attacks [10], [11], [12], such as adding perturbation to the observation, manipulating actions, and reward poison. Existing safe RL works assume a secure environment, and their training time vulnerability has barely been investigated in an adversarial setting. We believe that investigating such vulnerability of safe RL is important to enhance safety in the real world.

Conventional adversarial RL (nonsafe RL) methods focus on compromising the performance of DRL policies by reducing the cumulative reward [13], [14], [15]. They are not suitable for analyzing safety violations in safe RL, which has more serious consequences than reward reduction. We investigate whether a well-designed adversary could maliciously inject safety violation behavior into the learned policy. Specifically, we consider an adversary setting termed as "backdoor attack," in which the adversary injects the safety violation behavior (backdoor behavior) into the safe RL policy. The backdoor behavior will be triggered after the policy is deployed when some specific states are reached.

Considering the research gap, we study the vulnerability of safe RL during training. We focus on the formal language-guided safe RL especially the STL-guided safe RL, which converts the safety constraint and task specifications into a reward function. Unlike traditional DRL using hand-engineered reward function, STL effectively expresses the safety constraint and training the policy and is proven by several works [16], [17], [18].

In this article, we aim to address three key research questions: 1) How to design an effective backdoor attack that successfully compromises the control policy in terms of safety violation? 2) How does the effectiveness of an attack vary with different levels of its capability and knowledge? and 3) How to keep an attack effective while stealthy? To answer these questions, we formally define backdoor attacks for safe RL,

Manuscript received 13 August 2024; accepted 13 August 2024. This work was supported in part by NSF under Grant CNS-2333980. This article was presented at the International Conference on Embedded Software (EMSOFT) 2024 and appeared as part of the ESWEEK-TCAD special issue. This article was recommended by Associate Editor S. Dailey. *(Corresponding author: Fanxin Kong.)*

The authors are with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556 USA (e-mail: sjiang5@nd.edu; mliu9@nd.edu; fkong@nd.edu).

Digital Object Identifier 10.1109/TCAD.2024.3447468

then propose algorithms to synthesize such attacks, and finally validate the effectiveness of our methods theoretically and experimentally. To be specific, the main contributions of our paper are as follows.

1) We formally analyze the training time vulnerability of STL-guided safe RL and show that safe RL is unsafe when confronting a malicious adversary.
2) We define active and passive backdoor attacks, depending on whether to manipulate the observation, for safe RL. We propose two attack synthesis algorithms for each kind of attack, respectively, and theoretically show the correctness and effectiveness of our algorithms.
3) We perform extensive experiments on four benchmarks in OpenAI Safety Gym. The results show that our algorithms are effective in violating safety constraints while staying stealthy.

The remaining sections are organized as follows. Section II introduces the related work. Section III discusses necessary preliminary. In Section IV we introduce the proposed backdoor attack framework. Section V evaluates the proposed attack. Section VI discusses the limitations and defense. Section VII summarizes this article.

II. Related Work

This section discusses two major related works: 1) formal language (especially STL)-guided safe RL and 2) existing training time attacks targeted at reinforcement learning (RL).

A. Formal Language-Guided Safe RL

Formal languages, notably STL, offer a means to express control objectives and safety requirements. Specifically, these languages convert the desired system behavior into explicit specifications and ensure the system strictly adheres to these specifications [19]. Furthermore, [20] introduces robustness metrics to translate the boolean value of the STL specification into a real value. This approach efficiencies the process for STL-guided safe RL, eliminating the need for manual design of the reward function. Existing works [17], [21] show the efficacy of using the robustness metrics of STL to synthesize control policy. A recent work by Liu et al. [22] introduces the ASAP-Phi framework. This framework encourages the agent to fulfill the STL specification while minimizing the time taken to achieve it. Venkataraman et al. [23] focused on the computationally intractable problem where they propose a new state-space representation to capture the state history.

One significant line of research focuses on exploring the properties of robustness metrics and their impact on the learning process. Mehdipour et al. [24] were the first to propose the soundness property of robustness metrics, which rigorously classifies whether a trajectory satisfies the specification using values greater than 0 or less than 0. Building on this, Varnai and Dimarogonas [25] introduced the shadowing property of robustness metrics, highlighting its potential impact on learning efficiency. Another study by Singh and Saha [16] emphasizes the smoothness property and introduces a novel robustness metric aimed at maximizing smoothness, with the cost of sacrificing soundness. In our work, we utilize the robustness metrics introduced in [25], which are considered state-of-the-art methods for enhancing learning efficiency.

B. Training Time Attacks on RL

Training time adversarial attack means that a malicious adversary externally adds or manipulates the RL signals in the training phase, i.e., state, action, and reward so that the control policy is misled to act as the adversary's expectation [26], [27], [28], [29], [30]. While these attacks have shown impressive results in reducing the performance of the learning policy and decreasing the expected reward, they often lack stealthiness. In other words, the victim can easily detect that the policy is not functioning properly.

To address this, Panagiota et al. [13] proposed a backdoor attack on RL. They define a 3×3 patch in the corner of the image as the trigger. In this setup, the policy behaves as the standard policy when the patch is not presented, but it experiences a significant performance drop when the patch is presented. Gong et al. [31] considered the setting of offline RL and trigger the attack not only a patch on the image but also a particular system state (velocity). Additionally, [14] investigates the backdoor attack on competitive RL and they trigger the attack when one of the agents takes a specific action that leads to a fast-failing of the system. However, such works do not consider a major issue in designing the backdoor attack.

1) They lack a theoretical analysis of the adversary's reward design. Typically, when injecting malicious actions, they assign high positive rewards, which often require empirical knowledge and manual crafting.
2) None of the attacks consider a real-world scenario, where safety violations are much more critical than simply reducing the system's performance. Our work addresses these gaps, proposing backdoor attack algorithms aiming at safety violations with a theoretical reward design.

III. Preliminary

This section introduces the necessary preliminaries covered in this article. We briefly introduce STL and the STL-guided safe RL and present the system model and threat model.

A. Signal Temporal Logic

STL is a temporal logic designed to articulate various temporal properties using real-time signals. The STL specification is recursively constructed through subformulas and temporal operators. It yields either *true* or *false* based on a function $f:\mathbb{R}^n\rightarrow\mathbb{R}$ and can be inductively described by the following syntax:

$$\phi := \text{true}|\neg\varphi|\varphi_1\wedge\varphi_2|\mathbf{G}_{[a,b]}\varphi|\mathbf{F}_{[a,b]}\varphi \mid \varphi_1\mathbf{U}_{[a,b]}\varphi_2$$

where ϕ and φ are STL formulas. $\neg$ (negation) and $\wedge$ (conjunction) are Boolean operators. $\mathbf{G}$ (always), $\mathbf{F}$ (finally), and $\mathbf{U}$ (until) are temporal operators. The specification $\mathbf{G}_{[a,b]}\varphi$ is true if the property defined by φ is always true in the time horizon $[a,b]$. In addition, the $\mathbf{F}_{[a,b]}\varphi$ holds only if there is at least one time step where φ is true. Similarly, $\varphi_1\mathbf{U}_{[a,b]}\varphi_2$ is satisfied when φ_1 remains *true* until φ_2 becomes *true* during time horizon $[a,b]$.

The STL allows various definitions of robustness metrics to convert the boolean value into a real number to represent how satisfied the STL specification is. Based on this property, existing work [20] utilizes the robustness value as a reward function in RL so that they do not need to hand engineer the reward function. The robustness metrics are essential because the reward function (robustness metrics) significantly impacts learning an optimal RL policy. The original robustness metrics from [20] use min function to obtain the robustness of a conjunction operator and define the robustness metrics as follows:

$$\begin{aligned}
\rho(\mathbf{x}_t, \mu(\mathbf{x}_t) < d) &= d - \mu(\mathbf{x}_t)\\
\rho(\mathbf{x}_t, \neg\varphi) &= -\rho(\mathbf{x}_t, \varphi)\\
\rho(\mathbf{x}_t, \varphi_1 \wedge \varphi_2) &= \min(\rho(\mathbf{x}_t, \varphi_1), \rho(\mathbf{x}_t, \varphi_2))\\
\rho\big(\mathbf{x}_t, F_{[a,b]}\varphi\big) &= \max_{t' \in [a,b]} \rho(\mathbf{x}_{t'}, \varphi)\\
\rho\big(\mathbf{x}_t, G_{[a,b]}\varphi\big) &= \min_{t' \in [a,b]} \rho(\mathbf{x}_{t'}, \varphi)\\
\rho\big(\mathbf{x}_t, \varphi_1 \mathbf{U}_{[a,b]}\varphi_2\big) &= \max_{t \in [t+a, t+b]}\left(\min\left(\rho(\mathbf{x}_t, \varphi_2), \min_{t'' \in [t,t')} \rho(\mathbf{x}_{t''}, \varphi_1)\right)\right).
\end{aligned}$$

We denote the $\mathbf{x}_t$ is the state trajectory for the system that $\mathbf{x}_t = (x_0, x_1, \ldots, x_t)$.

However, these robustness metrics create a shadow-lifting problem that hurts the learning performance. The min function from the conjunction operator $\wedge$ allows increasing an individual specification without any impact on the overall robustness unless the specification's robustness is the minimum [25]. Instead, we consider state-of-the-art robustness metrics from [25] which solves the shadow-lifting problem and replaces the original min function from conjunction to the equation as follows:

$$\bar{\rho}_i = (\rho_i - \rho_{\min})/\rho_{\min}$$

$$\rho(\mathbf{x}_t, (\rho_1 \wedge \rho_2 \ldots \wedge \rho_n)) = \begin{cases} \frac{\sum_i \rho_{\min} e^{\bar{\rho}_i} e^{\nu \bar{\rho}_i}}{\sum_i e^{\nu \bar{\rho}_i}}, & \text{if } \rho_{\min} < 0\\ \frac{\sum_i \rho_i e^{-\nu \bar{\rho}_i}}{\sum_i e^{-\nu \bar{\rho}_i}}, & \text{if } \rho_{\min} > 0\\ 0, & \text{if } \rho_{\min} = 0. \end{cases} \quad (1)$$

We denote $\rho_{\min}$ as the robustness value of which ρ_i achieves the minimum among all subspecification φ and ν is a hyperparameter defined by the user.

Although the most recent work by Singh and Saha [16] proposes a new semantics that yields the best performance in learning STL-guided control policies, we use the approach outlined in Varnai and Dimarogonas [25] for learning the control policies. Our focus is to explore the vulnerability of STL-guided control policy instead of improving learning efficiency; hence, different robustness metrics do not impact the theoretical proof.

B. System Model

In this article, we investigate the safety vulnerability of CPS. We assume that the CPS with unknown system dynamics has a specific task to complete (goal) within a time horizon T. Additionally, several unsafe regions need to be avoided, meaning certain states should not be reached (safety constraint). For example, an autonomous vehicle aims to reach a target position while needing to avoid collisions with obstacles and other vehicles. Similarly, a robot arm strives to grasp a box while avoiding contact with other objects. We formally define the goal and safety constraint using STL.

Definition 1 (Goal): We denote the STL specification φ_g to be the goal of the system. Given the start time t_0 and a time horizon T, the system achieves the goal (complete the task) only if $\rho(\mathbf{x}_t, F_{[t_0, t_0+T]}\varphi_g) \geq 0$.

Definition 2 (Safety Constraint): We denote the STL specification φ_s to be the safety constraint. Given the start time t_0 and a time horizon T, the system satisfies the safety constraint (avoid unsafe) only if $\rho(\mathbf{x}_t, G_{[t_0, t_0+T]}\varphi_s) \geq 0$.

The system aims to simultaneously achieve the goal and satisfy the safety constraint by interacting with the environment. Combining the STL specification of goal and safety constraint, the overall STL specification is

$$\phi = F_{[t_0, t_0+T]}\varphi_g \wedge G_{[t_0, t_0+T]}\varphi_s. \quad (2)$$

Note that obtaining the actual states of a real-world CPS is challenging. Instead, we assume that the system relies on sensor values (observations) to determine its state. Throughout this article, we consider the sensor values (observations) at time step t as the system state x_t.

C. STL-Guided Safe RL

We assume the system tries to find a control policy π that maximizes the robustness of ϕ. We formulate a safe learning process that utilizes the STL specification.

Definition 3: The safe learning process for a safety-critical system can be formulated as a finite-horizon constraint Markov decision process (CMDP) defined as a tuple $Q := (S, A, T, p, r, c, \gamma)$, where S and A are the state and action space, respectively; T is the total time steps that the system interacts with the environment; p is the transition function that $p : S \times A \times S \rightarrow [0, 1]$ and $p(x_t, a, x_{t+1})$ is the probability that taking an action $a \in A$ at state $x_t \in S$ and result in the next state x_{t+1}; and r, c, and γ are the reward function, cost function, and discount parameter, respectively.

The objective of STL-guided safe RL is to obtain an optimal control policy $\pi : S \rightarrow A$ that can maximize the cumulative reward by using the robustness metric as the reward function

$$\pi = \arg\max_{\pi} \mathbb{E}^{\pi} \sum_{t=0}^{T} \gamma^t \rho(\mathbf{x}_t, \phi).$$

In this article, we assume the systems employing actor–critic algorithms [32] for safe RL. Actor–critic algorithms have demonstrated efficiency in addressing continuous learning problems and are recognized for their sample efficiency, leveraging the critic network for Q function approximation, also known as the state–action value. We show the Q function and the value function V in the STL-guided RL as follows:

$$\begin{aligned}
Q^{\pi}(x_t, a_t) &= \rho(\mathbf{x}_t, \phi) + \gamma \max_{a_{t+1}} Q^{\pi}(x_{t+1} | (x_t, a_t), a_{t+1})\\
V^{\pi}(x_t) &= , \sum_{k=0}^{T} \gamma^k \rho(\mathbf{x}_t, \phi). \quad (3)
\end{aligned}$$

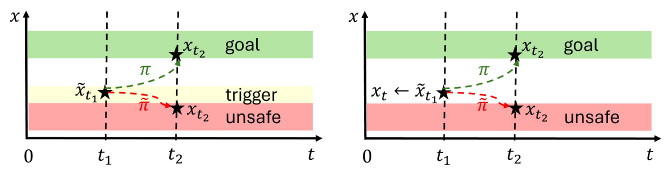

Fig. 1. Illustration of passive backdoor attack (left) and active backdoor attack (right).

TABLE I
STRONG ATTACK CAN MANIPULATE THE ACTION SIGNAL DURING TRAINING, WHEREAS THE WEAK ATTACK CANNOT. IN THE CASE OF THE ACTIVE BACKDOOR ATTACK, THE STATE x_t IS MANIPULATED TO CREATE THE TRIGGER. ON THE OTHER HAND, THE PASSIVE BACKDOOR ATTACK DOES NOT REQUIRE THE MANIPULATION OF THE STATE

Attack	Strong	Weak
Active Backdoor	x_t, a_t, r_t	x_t, r_t
Passive Backdoor	a_t, r_t	r_t

D. Threat Model

In this article, we consider a malicious adversary that can access the training process of the safe RL. We refer to a policy trained using STL-guided safe RL as the "standard policy," and one tampered with by the adversary as the "Trojaned policy." We first illustrate the adversary's knowledge and capability.

Adversary Knowledge: We assume that the adversary has complete access to the sensor data (state) and the STL specification ϕ used in the training process. For an adversary executing a strong attack, as defined in Section IV, this includes having knowledge of the system and its environment. However, the adversary lacks knowledge of the RL algorithm and cannot access its parameters.

Adversary Capability: The adversary can manipulate both the sensor data and reward signal during the training phase. Furthermore, if the adversary can manipulate the action signal, we categorize it as a *strong attack*; otherwise, it is termed a *weak attack*, as defined in this work [13].

Rather than compelling the system to learn a minimally performing control policy, we consider a more severe scenario wherein the control policy should operate normally unless certain states trigger a violation. This approach poses greater risk as it may allow the system to overlook vulnerabilities prior to deployment. Essentially, the Trojaned policy could potentially produce actions that violate safety constraints when encountering specific states but operate normally otherwise. This strategy is referred to as a *backdoor attack*.

Definition 4 (Backdoor Attack and Backdoor Behavior): Suppose for a set of state (observation) space $\tilde{S}$, a Trojaned policy $\tilde{\pi} : S \rightarrow A$, for an initial state $x_0 \in \tilde{S}$, the Trojaned policy will result in a sequence of action $\tilde{a}_0\tilde{a}_1 \ldots \tilde{a}_t$ and a final state x_t which violates the safety constraint $\rho(\mathbf{x}_t | x_0, G_{[t_0,t_0+T]}\varphi_s) < 0$. We define the state space $\tilde{S}$ as the backdoor trigger and the sequence of action as backdoor behavior.

Adversary Objective: The adversary's objective is to inject the backdoor behavior into the control policy. In other words, the system leads to a safety violation and does not complete the goal when the trigger is presented. Meanwhile, the adversary should keep stealthy, that is, when the trigger is not presented, the control policy should work normally as the standard safe RL policy.

IV. BACKDOOR ATTACK DESIGN

Backdoor attacks on images typically involve creating a patch as the trigger for initiating the attack. Following this philosophy, we introduce the active backdoor attack, which manipulates the states as the triggers in the context of CPS. Additionally, we propose the passive backdoor attack, which does not require manipulating states for the triggers. Note that the Trojaned policy should work normally as a standard policy when no trigger is presented but lead to a safety violation behavior when the trigger is presented. Based on these requirements, we define the active backdoor attack and the passive backdoor attack.

Definition 5 (Active Backdoor Attack): We consider the active backdoor attack where the backdoor triggers are not in the original state space: $\tilde{S} \notin S$. The attack is triggered only if the adversary manipulates the observations $x_t \rightarrow \tilde{x}_t$.

The active backdoor attack follows the traditional backdoor attack strategy from existing work [13]. Instead of creating a fixed patch on the image as the trigger, the trigger for the CPS would be manipulating the observation with a fixed 'patch' with the adversary's selection. For instance, consider an autonomous vehicle equipped with an inertial measurement unit (IMU) sensor, capable of measuring linear velocity, angular velocity, and acceleration along the x, y, and z axes. The adversary can select the trigger and manipulate the values of less crucial sensors, such as the linear acceleration along the z-axis (representing gravity). We assume that this sensor data is deemed unimportant for autonomous driving tasks, and the system may overlook such biased sensor data, resulting in something bad happening. This active backdoor attack is triggered when the adversary manipulates the state as the trigger. Conversely, we propose a passive backdoor attack that does not require state manipulation.

Definition 6 (Passive Backdoor Attack): The passive backdoor attack is defined as the backdoor trigger belonging to the original state space: $\tilde{S} \in S$. This implies that the attack is triggered in some specific states even when the adversary does not manipulate the observations.

The passive backdoor attack involves the adversary selecting a set of normal states as triggers. The underlying concept is that the system operates normally; however, specific cases, such as nearing an unsafe region, trigger the attack. For instance, consider a robot arm equipped with a LiDAR sensor that detects the distance to surrounding objects. In a passive backdoor attack, the adversary might inject a backdoor behavior: when the LiDAR recognizes the robot arm nearing a wall, the Trojaned policy forces the robot arm to collide with the wall. Notably, in the passive backdoor attack, the adversary does not need to manipulate the observations to trigger the attack. This attack is deemed more harmful, as it provides the system with less time to react effectively.

As illustrated in the previous section, attacks are also categorized as strong or weak based on whether the adversary can modify the action as inspired by [13]. Table I classifies the attacks based on the adversary's strategy and capability, also indicating which signal the attack will manipulate. For example, a strong active backdoor attack can manipulate the state, action, and reward in a training phase while a weak active backdoor can only manipulate the state and reward. In the next section, we discuss how the adversary realizes the backdoor attack target on the STL-guided safe RL.

A. *Problem Formulation*

Intuitively, the adversary aims to have a Trojaned policy that generates action $\tilde{a}_t$ (approach to the unsafe) when the trigger is presented $\tilde{x}_t \in \tilde{S}$ while maintaining normal behavior when $x_t \notin \tilde{S}$. We denote a'_t as the optimal malicious action that leads the system to the unsafe region. Then ideally the adversary's goal is to have the Trojaned policy that

$$\tilde{\pi}(\tilde{x}_t) = a'_t \neq \pi(x_t)$$
$$\tilde{\pi}(x_t) = \pi(x_t) \neq a'_t$$

where π denotes the standard policy and $\tilde{\pi}$ denotes the Trojaned policy. The above equations demonstrate that the Trojaned policy normally acts as the standard policy with state x_t and performs the optimal malicious action a'_t when the trigger $\tilde{x}_t$ presents. Note that we use $\tilde{x}_t$ to denote the trigger state no matter whether it is a passive or active backdoor attack.

To better illustrate how the Trojaned policy works, we start from the state–action value (Q) function. The state–action value function used in RL expresses the expected reward if it takes action a_t at the state x_t. A higher value of the Q implies the control policy has a higher potential to take the action a_t. We show the state–action value function of the standard policy

$$\begin{aligned} &Q^{\pi}(\tilde{x}_t, a_t) > Q^{\pi}\left(\tilde{x}_t, a'_t\right) \\ &Q^{\pi}(x_t, a_t) > Q^{\pi}\left(x_t, a'_t\right) \\ &\text{where } a_t = \pi(\cdot). \end{aligned} \tag{4}$$

This expresses that the standard policy consistently prioritizes action a_t over a'_t as the latter may lead to safety violations, regardless of whether the state is the trigger state. However, the adversary has the opposite objective. We formulate the attack effectiveness as

$$Q^{\tilde{\pi}}(\tilde{x}_t, a_t) < Q^{\tilde{\pi}}\left(\tilde{x}_t, a'_t\right). \tag{5}$$

Equation (5) implies that the Trojaned policy will opt for the malicious action a'_t when the trigger is presented because it has the highest state–action value. Similarly, if the trigger is not presented, the state–action value should satisfy as follows:

$$Q^{\tilde{\pi}}(x_t, a_t) > Q^{\tilde{\pi}}\left(x_t, a'_t\right). \tag{6}$$

Equation (6) indicates that when the trigger is not presented, the Trojaned policy should output the action that does not aim at safety violation. We define the fulfillment of (6) as the attack being stealthy. In other words, the Trojaned policy is stealthy when it behaves as standard policies to fulfill the system's goal when no trigger is presented. We evaluate the stealthiness by comparing the difference between the Trojaned and standard policies in Section V.

Algorithm 1: Passive Backdoor Attack

Input : A victim policy π, the maximum length of trajectory T.
Output: Trojaned policy $\tilde{\pi}$

1 *step* $\leftarrow 0$;
2 **while** *step* $<$ *total_attack_steps* **do**
3 $t \leftarrow 0$
4 **for** $t < T$ **do**
5   Sample state x_t and trajectory $\mathbf{x}_t$
6   Sample $a_t = \pi(x_t)$
7   **if** $x_t \in \tilde{S}$ **then**
8   *step* $\leftarrow$ *step* $+ 1$
9   **if** *Attack is Strong Attack* **then**
10     $a_t \leftarrow$ malicious action a'_t
11   **end**
12   **end**
13   Sample x_{t+1} and trajectory $\mathbf{x}_{t+1}$
14   $r_t \leftarrow$ reward_poisoning$(\mathbf{x}_{t+1})$
15 **end**
16 Update policy π
17 **end**
18 Return policy π

Based on (4) and (5), we denote the r_p as a positive constant that the adversary uses to poison the reward, aiming to reduce $Q^{\pi}(\tilde{x}_t, a_t)$ and satisfy the following equation:

$$Q^{\pi}(\tilde{x}_t, a_t) - r_p < Q^{\pi}\left(\tilde{x}_t, a'_t\right). \tag{7}$$

In summary, the adversary's objective is to satisfy both (5) and (6), which represent the attack's effectiveness and stealthiness, respectively. However, both objectives are counter to the goal of safe RL learning, underscoring the importance of a well-designed attack.

B. *Passive Backdoor Attack*

In this section, we propose our passive backdoor attack algorithm. To fulfill (5) and (6), it is crucial to design a specific reward-poisoning method (i.e., manipulating the reward values). Unlike existing backdoor attacks on RL [13], which simply changes the reward to -1 or 1, offering nonguaranteed attack performance, we present the passive backdoor attack algorithm in Algorithm 1 and provide theoretical analysis of the algorithm.

Algorithm 1 details the passive backdoor attack procedure. The attack begins by checking if the current state is within a predetermined trigger space (close to unsafe regions). If so, the adversary modifies the action from a_t to $\tilde{a}_t$ for a strong attack, as outlined in lines 9–11. For a weak attack, the manipulation focuses solely on altering the reward, following the steps described in Algorithm 2 for reward poisoning. This involves using a positive constant r_p to decrease the reward r_t when transitioning between specific states ($\tilde{S}$ to S), as seen in lines 2–4, ensuring the attack remains effective yet stealthy. We will further demonstrate how the reward poisoning aligns with the goals of effectiveness and stealthiness, as outlined in (5) and (6), in the followings. When the system is near

Algorithm 2: Reward Poisoning

Input : Trajectory $\mathbf{x}_{t+1}$, original reward r_t, the reward penalty r_p.
Output: Reward r_t.

```
1  r_t ← ρ(x_{t+1}, φ)
2  if x_t ∉ S̃ and x_{t+1} ∈ S̃ then
3  |  r_t ← r_t − r_p
4  end
5  else if x_t ∈ S̃ and x_{t+1} ∈ S̃ then
6  |  r_t ← ρ(x_{t+1}, φ′)
7  end
8  else if the system violates the safety then
9  |  r_t = r_p
10 end
11 return r_t
```

unsafe states, the altered rewards prompt the learning policy toward safety violations, detailed in lines 5–7, leveraging ϕ' to define the adversary's desired outcome based on the STL specification. The specification ϕ' is given as follows:

$$\phi' = G_{[t_0,t_0+T]}\neg\varphi_g \wedge F_{[t_0,t_0+T]}\neg\varphi_s.$$

The ϕ' represents the adversary goal which is opposite to the original system specification ϕ. It is designed such that $G_{[t_0,t_0+T]}\neg\varphi_g$ signifies the adversary's intention for the system to consistently avoid the goal within a specified timeframe. Similarly, $F_{[t_0,t_0+T]}\neg\varphi_s$ encourages the system to engage in behavior that violates safety protocols. Furthermore, as detailed in lines 8–10 of Algorithm 2, when the system is already in a state of safety violation, the adversary assigns a positive reward r_p. This strategy is employed to enhance the likelihood of the policy thereby maximizing the attack's effectiveness.

To summarize, the passive backdoor attack remains inactive while the system is far from any unsafe areas. The attack begins once the system nears an unsafe zone. Initially, to ensure stealth, the system's reward is reduced by r_p when entering the trigger states. This penalty discourages the system from approaching unsafe areas from a long distance. However, if the system is inside the trigger states, the adversary then incentivizes this behavior by rewarding the system based on the robustness of ϕ' and further offers a final bonus of r_p if the system violates safety. This approach subtly encourages safety violation actions only when the system is close to unsafe regions, aligning to make the backdoor attack stealthy.

Theorem 1: Assume Θ is the minimum robustness of a trigger state $\tilde{x}_t \in \tilde{S}$ denote as $\Theta := \min_{\tilde{x}_t\in\tilde{S}}\rho(\mathbf{x}_t, \phi)$ and it is easy to have $\Theta < 0$. Suppose (7) holds for the policy π, the lower bound of the r_p to satisfy the effectiveness and stealthiness is given by

$$r_p > \frac{\gamma}{\gamma - 1}\Theta. \tag{8}$$

Theorem 1 establishes the minimum value for r_p, guiding its selection to maintain the stealthiness of the backdoor attack. The proof of Theorem 1 is presented as follows.

Proof: From (7), we have

$$r_p > Q^{\pi}(\tilde{x}_t, a_t) - Q^{\pi}\left(\tilde{x}_t, a_t'\right).$$

We derive the upper bound for the difference between the Q-values of the original and manipulated actions at state $\tilde{x}_t$ as follows:

$$Q^{\pi}(\tilde{x}_t, a_t) - Q^{\pi}(\tilde{x}_t, \tilde{a}_t) \leq \max_{\tilde{x}_t\in\tilde{S}} Q^{\pi}(\tilde{x}_t, a_t) - \min_{\tilde{x}_t\in\tilde{S}} Q^{\pi}(\tilde{x}_t, \tilde{a}_t).$$

To evaluate the right-hand side of the equation, we introduce Lemma 1 for calculating $\max Q^{\pi}(\tilde{x}_t, a_t)$. ■

Lemma 1: Suppose the trajectory $\mathbf{x}_t$ with an initial state $\tilde{x}_0 \in \tilde{S}$, the maximum Q value the state $\tilde{x}_0$ achieve will be

$$\max_{\tilde{x}_t\in\tilde{S}} Q^{\pi}(\tilde{x}_t, a_t) \leq 0.$$

Proof: We have

$$Q^{\pi}(\tilde{x}_t, a_t) = \rho(\mathbf{x}_t, \phi) + \gamma Q^{\pi}(x_{t+1}, a_{t+1}).$$

The trajectory with a final state $\tilde{x}_t$ does not satisfy the STL specification φ_g. According to the definition of soundness [25], we have $\rho(\mathbf{x}_t, \phi) < 0$. Similarly, for any trajectory $\mathbf{x}_t$ that does not satisfy the goal, its robustness value is less than 0. We can easily have the upper bound of $Q^{\pi}(\tilde{x}_t, a_t) \leq 0$. ■

We then introduce Lemma 2 to determine the bounded value of $\min Q^{\pi}(\tilde{x}_t, \tilde{a}_t)$.

Lemma 2: Given the minimum robustness among all states in the trajectory Θ and a Q function with a state $\tilde{x}_0 \in \tilde{S}$ and action $\tilde{a}_t$, we have

$$\min_{\tilde{x}_t\in\tilde{S}} Q^{\pi}(\tilde{x}_t, \tilde{a}_t) \geq \frac{\gamma}{1-\gamma}\Theta.$$

Proof: Lemma 2 gives a lower bound of the $Q^{\pi}(\tilde{x}_t, \tilde{a}_t)$. We prove this by assuming a minimum robustness value Θ, where Θ is the minimum robustness value in the trigger space, denoted as $\Theta = \min_{\tilde{x}\in\tilde{S}}(\rho(\mathbf{x}_t, \phi))$

$$\begin{aligned}
\min_{\tilde{x}_t\in\tilde{S}} Q^{\pi}(\tilde{x}_t, \tilde{a}_t) &= \rho(\mathbf{x}_t, \phi) + \gamma Q^{\pi}(x_{t+1}, a_{t+1}) \\
\min_{\tilde{x}_t\in\tilde{S}} Q^{\pi}(\tilde{x}_t, \tilde{a}_t) &\geq \Theta + \gamma\Theta + \gamma^2\Theta \cdots + \gamma^{T-t}\Theta \\
&\geq \frac{\gamma}{1-\gamma}\Theta.
\end{aligned}$$

■

Based on Lemmas 1 and 2, we can have the lower bound of r_p

$$r_p > \frac{\gamma}{\gamma - 1}\Theta.$$

■

The lower bound of r_p is only related to the discount factor γ and minimum robustness value Θ, both of which can be predicted or acquired by the adversary. For example, the γ is usually set to 0.99 in the RL training. The Θ can be obtained by sampling the training data and monitoring the robustness value.

Algorithm 3: Active Backdoor Attack

Input : A victim policy π, the maximum length of trajectory T, selected trigger $\tilde{x}$.
Output: Trojaned policy $\tilde{\pi}$

1 *step* $\leftarrow 0$;
2 **while** *step* $<$ *total_attack_steps* **do**
3 $\quad t \leftarrow 0$
4 $\quad$ **for** $t < T$ **do**
5 $\quad\quad$ Sample state x_t and trajectory $\mathbf{x}_t$
6 $\quad\quad$ Sample $a_t = \pi(x_t)$
7 $\quad\quad$ **if** *time to attack* **then**
8 $\quad\quad\quad x_t \leftarrow \tilde{x}_t$
9 $\quad\quad\quad$ *step* $\leftarrow$ *step* $+ 1$
10 $\quad\quad\quad$ **if** *Attack is Strong Attack* **then**
11 $\quad\quad\quad\quad a_t \leftarrow$ malicious action $\tilde{a}_t$
12 $\quad\quad\quad$ **end**
13 $\quad\quad$ **end**
14 $\quad\quad$ Sample x_{t+1} and trajectory $\mathbf{x}_{t+1}$
15 $\quad\quad r_t \leftarrow$ reward_poisoning($\mathbf{x}_{t+1}$)
16 $\quad$ **end**
17 $\quad$ Update policy π
18 **end**
19 Return policy π

C. Active Backdoor Attack

The active backdoor attack requires the adversary to designate certain states as triggers by introducing perturbations, as detailed in Algorithm 3. The process begins with the adversary selecting specific states to perturb, setting them as triggers for the attack. During the training phase, the adversary then manipulates the process by altering states, rewards, and in cases of a strong attack, actions too, as demonstrated in lines 7–15. Furthermore, to ensure both effectiveness and stealthiness, the active backdoor attack involves adjusting the state-value function when the system transitions from a nontrigger to a trigger state. A key distinction of the active backdoor attack, compared to the passive backdoor, is that the Trojaned policy behaves normally without manipulating the states, becoming effective only when the adversary introduces poisoned states as triggers.

While the active backdoor attack can be triggered at any time by manipulating the states to triggers, the adversary may strategically choose to trigger the attack when the system is near unsafe states during the training phase. This aims to maximize the likelihood of safety violations and enhance the efficiency of the attack.

From the perspective of the victim system, the Trojaned policy created by the active backdoor attack is stealthier compared to the passive backdoor. This is because the active backdoor remains hidden until the adversary decides to trigger the attack. However, the active backdoor requires the adversary to manually perturb the states to trigger the attack, while the passive backdoor can automatically start the safety violation.

We introduce Proposition 1 to prove that our active and passive backdoor attacks satisfy the effectiveness and stealthiness.

Proposition 1: The Trojaned policy $\tilde{\pi}$ produced by the passive and active backdoor attack fulfills the requirements of attack effectiveness and stealthiness, as specified in (5) and (6).

Proof: We start by demonstrating that the Trojaned policy $\tilde{\pi}$, resulting from the backdoor attack, is effective

$$Q^{\tilde{\pi}}(\tilde{x}_t, \tilde{a}_t) - Q^{\tilde{\pi}}(\tilde{x}_t, a_t) = V^{\tilde{\pi}}(x_{t+1}|(\tilde{x}_t, \tilde{a}_t)) - V^{\tilde{\pi}}(x_{t+1}|(\tilde{x}_t, a_t)).$$

Assume the new stat $(x_{t+1}|(\tilde{x}_t, a_t))$ is still in the trigger space. According to Algorithms 1 and 2, we have the value function

$$V^{\tilde{\pi}}(x_{t+1}|(\tilde{x}_t, \tilde{a}_t)) = \mathbb{E}^{\pi} \sum_{k=0}^{t_1-1} \gamma^k \rho\big(\mathbf{x}_{t+1+k}|(\tilde{x}_t, \tilde{a}_t), \phi'\big) + \gamma^{t_1} r_p$$
$$V^{\tilde{\pi}}(x_{t+1}|(\tilde{x}_t, a_t)) = \mathbb{E}^{\pi} \sum_{k=0}^{t_2-1} \gamma^k \rho\big(\mathbf{x}_{t+1+k}|(\tilde{x}_t, a_t), \phi'\big) + \gamma^{t_2} r_p.$$

We define t_1 and t_2 are the number of time steps until the system violates the safety. Where $\tilde{a}_t$ is the optimal malicious action that maximizes the robustness value of ϕ', so we can easily have

$$V^{\tilde{\pi}}(x_{t+1}|(\tilde{x}_t, \tilde{a}_t)) > V^{\tilde{\pi}}(x_{t+1}|(\tilde{x}_t, a_t))$$
$$Q^{\tilde{\pi}}(\tilde{x}_t, \tilde{a}_t) > Q^{\tilde{\pi}}(\tilde{x}_t, a_t).$$

If the $(x_{t+1}|(\tilde{x}_t, a_t))$ is not in the trigger space, we have

$$V^{\tilde{\pi}}(x_{t+1}|(\tilde{x}_t, a_t)) = \mathbb{E}^{\pi} \sum_{k=1}^{T-t-1} \gamma^k \rho(\mathbf{x}_{t+k}|(\tilde{x}_t, a_t), \phi).$$

Based on Theorem 1, we have that $V^{\tilde{\pi}}(x_{t+1}|(\tilde{x}_t, \tilde{a}_t)) > V^{\tilde{\pi}}(x_{t+1}|(\tilde{x}_t, a_t))$ holds. So the Trojaned policy $\tilde{\pi}$ satisfies the effectiveness.

Similarly, we have the Q-function for the stealthiness

$$Q^{\tilde{\pi}}(x_t, a_t) - Q^{\tilde{\pi}}\big(x_t, a_t'\big) = V^{\tilde{\pi}}(x_{t+1}|(x_t, a_t)) - V^{\tilde{\pi}}\big(x_{t+1}|\big(x_t, a_t'\big)\big).$$

Suppose $(x_{t+1}|(x_t, a_t'))$ still does not belong to the trigger space, we have the value function of the x_{t+1}

$$V^{\tilde{\pi}}(x_{t+1}|(x_t, a_t)) = \mathbb{E}^{\pi} \sum_{k=1}^{T-t-1} \gamma^k \rho(\mathbf{x}_{t+k}|(x_t, a_t), \phi)$$
$$V^{\tilde{\pi}}\big(x_{t+1}|\big(x_t, a_t'\big)\big) = \mathbb{E}^{\pi} \sum_{k=1}^{T-t-1} \gamma^k \rho\big(\mathbf{x}_{t+k}|\big(x_t, a_t'\big), \phi\big).$$

While a_t is the optimal action that maximize the robustness of ϕ, we have

$$V^{\tilde{\pi}}(x_{t+1}|(x_t, a_t)) > V^{\tilde{\pi}}\big(x_{t+1}|\big(x_t, a_t'\big)\big)$$
$$Q^{\tilde{\pi}}(x_t, a_t) > Q^{\tilde{\pi}}\big(x_t, a_t'\big).$$

If $(x_{t+1}|(x_t, a_t'))$ goes into the trigger space, then the system will lead to safety violation. We have

$$V^{\tilde{\pi}}\big(x_{t+1}|(x_t, a_t')\big) = \mathbb{E}^{\pi} \sum_{k=1}^{t_1-1} \gamma^k \rho\big(\mathbf{x}_{t+k}|\big(x_t, a_t'\big), \phi'\big) - r_p + \gamma^{t_1} r_p.$$

We have

$$\mathbb{E}^{\pi} \sum_{k=1}^{T-t-1} \gamma^k \rho(\mathbf{x}_{t+k}|(x_t, a_t), \phi) - \mathbb{E}^{\pi} \sum_{k=1}^{t_1-1} \gamma^k \rho\big(\mathbf{x}_{t+k}|\big(x_t, a_t'\big), \phi'\big)$$
$$> \gamma^{t_1} r_p - r_p.$$

Note that t_1 denotes the number of time steps from t until the system violates safety. This implies that for a sufficiently

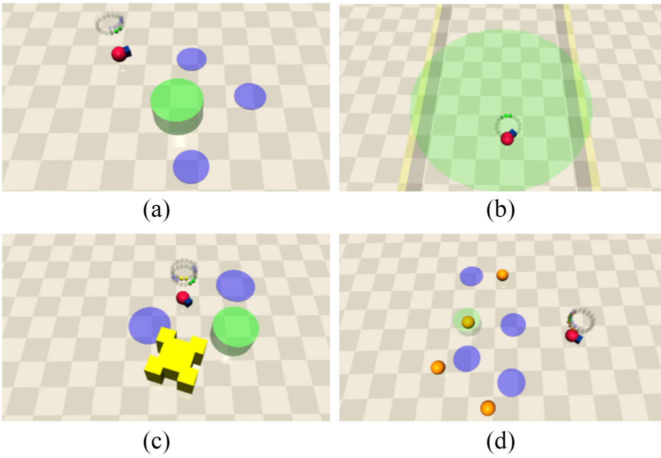

Fig. 2. Four benchmarks used in our experiments from Safety Gym. (a) Goal. (b) Circle. (c) Push. (d) Button.

large t_1, the manipulated policy $\tilde{\pi}$ satisfies the stealthiness criterion. Moreover, a larger value of r_p increases the absolute value of $\gamma^{t_1} r_p - r_p$, which in turn enhances the likelihood of fulfilling the stealthiness requirements. This conclusion is in line with Theorem 1. ■

V. Experiments

This section demonstrates our experimental approach for assessing the effectiveness of our backdoor attack on different benchmarks. All experiments were carried out on a system featuring an Intel Core i7-13700F processor operating at 2.10 GHz with 16 cores and 16 GB of RAM.

A. Benchmarks

Safety Gym: We implement our attack algorithms on the OpenAI Safety Gym [33], [34]. The Safety Gym offers safe RL benchmarks to address the challenge of safe exploration. We focus specifically on the Goal, Circle, Push, and Button benchmarks and use the Point agent to represent the victim system.

The Goal benchmark is a typically reach-avoid problem in which a point navigates to a green goal while avoiding contact with the three unsafe hazards on the map. The PointGoal benchmark can be formulated into STL specification as follows:

$$\phi = F_{[0,T]}(d_g < r_g) \wedge G_{[0,T]}(d_c > r_c)$$

where d_g is the distance to the goal and d_c is the distance to the closest hazard.

The Circle benchmark requires the point to navigate in the green circle while avoiding going outside the boundaries where the point has 16 sensors to detect the distance to the center of the circle. Meanwhile, two walls are on the two sides so the car should not crash on the wall. The goal of the point is to reach a high velocity inside the circle and the safety constraint is not crashing into the wall. We formulate the goal and safety constraint as follows:

$$\phi = F_{[0,T]}\left(\frac{v}{|r_{car} - r_{circle}|} > v_0\right) \wedge G_{[0,T]}(d_c > 0).$$

We denote that v is the current velocity of the car and v_0 is the desired velocity. r_{car} denotes the distance from the car to the center of the circle which encourages the car to navigate away from the center but not going out of the circle.

The Push benchmark adds a yellow box compared to the Goal benchmark. In this scenario, the Point must push the box to the goal while avoiding two hazards. The STL specification for this benchmark is

$$\phi = F_{[0,T]}(d_g < r_g) \wedge F_{[0,T]}(d_b < r_b) \wedge G_{[0,T]}(d_c > r_c).$$

Here, d_g represents the box-to-goal distance, d_b is the point-to-box distance, and d_c is the distance to the hazards, with r_g, r_b, and r_c being the respective thresholds.

The Button benchmark presents a similar reach-avoid problem, where the Point must touch the correct button while avoiding hazards and the wrong button. The STL specification is the same as the Goal benchmark with additional $G_{[0,T]}(d_w > r_w)$ to avoid the wrong button

$$\phi = F_{[0,T]}(d_g < r_g) \wedge G_{[0,T]}(d_w > r_w) \wedge G_{[0,T]}(d_c > r_c).$$

B. Experiments Setting

Training Setting: We employ the proximal policy optimization (PPO) algorithm [35] to train the control policy across four benchmarks, utilizing 10^7 training steps. The discount factor γ is set to 0.99 to balance immediate and future rewards. The architecture of the control policy comprises a three-layer fully connected NN, utilizing the rectified linear unit (ReLU) activation function.

Adversary Setting: We conduct the four backdoor attacks and use SP, WP, SA, and WA to denote strong passive, weak passive, strong active, and weak active, respectively. Then We define ϵ as the fraction representing how much of the training process can be interfered with by an adversary, with values set at 0.005, 0.01, 0.015, and 0.02. These values indicate the maximum proportion of the training steps that can be poisoned. The steps to be poisoned with are chosen randomly, and once the amount of poisoned steps reaches the specified fraction, the adversary is not allowed to manipulate any further in the training process.

In the case of the active backdoor attack, the trigger condition is met by modifying the acceleration sensor's reading to a value of 5. Conversely, the passive backdoor attack is initiated when the system comes within 0.3 distance units of an unsafe region. For both types of attacks across all four benchmarks, we employ a reward penalty value, $r_p = 50$, as illustrated in Algorithm 2. This value of $r_p = 50$ is considered sufficiently large for the context of these benchmarks and aligns with the recommendations posited in Theorem 1.

Baseline Settings: We conduct a comparative analysis between our backdoor attack and baselines [31] and [13]. Both baselines utilize the idea of poisoning states and rewards during attacks and poisoning actions during strong attacks. We implement both strong (ST) and weak (WT) versions of the baselines using the same trigger as our active backdoor attack. For the reward poisoning setting in [13], we assign $r_t = +1$ during strong attacks. The original weak baseline's reward mechanism is tailored for discrete action spaces, which

TABLE II
EFFECTIVENESS OF THE BACKDOOR ATTACK IS EVALUATED THROUGH THE VIOLATION RATES, WITH ϵ REPRESENTING THE RATIO OF POISONED TRAINING STEPS. WE USE ABBREVIATIONS TO DENOTE DIFFERENT ATTACK SCENARIOS: SP AND WP REFER TO THE PROPOSED STRONG PASSIVE AND WEAK PASSIVE BACKDOOR ATTACKS, WHILE SA AND WA REPRESENT STRONG ACTIVE AND WEAK ACTIVE BACKDOOR ATTACKS, RESPECTIVELY. ADDITIONALLY, ST AND WT DENOTE THE BASELINE METHODS OF STRONG TROJANDRL AND WEAK TROJANDRL

ϵ	Goal						Circle					
	SP	**WP**	**SA**	**WA**	ST	WT	**SP**	**WP**	**SA**	**WA**	ST	WT
0.005	28.9% ±4.3%	24.8% ±3.3%	12.1% ±1.8%	8.4% ±3.4%	10.8% ±2.7%	14.1% ±5.0%	16.6% ±3.2%	12.8% ±4.4%	15.9% ±3.0%	14.4% ±5.0%	5.4% ±1.8%	2.2% ±0.3%
0.01	42.0% ±2.1%	35.9% ±6.3%	41.0% ±4.8%	24.6% ±3.5%	17.0% ±4.2%	22.0% ±1.8%	20.6% ±2.2%	20.0% ±2.6%	26.8% ±4.3%	18.6% ±2.6%	11.6% ±3.5%	10.1% ±2.0%
0.015	54.2% ±2.8%	38.7% ±3.8%	48.6% ±5.7%	45.8% ±6.9%	21.2% ±3.5%	17.6% ±3.6%	28.7% ±5.2%	23.9% ±2.4%	25.8% ±3.0%	19.3% ±2.8%	10.4% ±1.8%	11.0% ±3.3%
0.02	51.4% ±5.0%	41.2% ±5.6%	60.0% ±5.1%	50.6% ±1.3%	35.6% ±3.7%	33.0% ±1.2%	36.8% ±5%	28.8% ±1.9%	49.6% ±4.7%	40.8% ±4.2%	9.6% ±1.8%	10.8% ±4.2%
	Push						Button					
0.005	85.6% ±3.9%	48.4% ±6.1%	64.3% ±6.3%	38.2% ±5.4%	37.4% ±3.3%	20.2% ±3.0%	48.2% ±4.4%	33.8% ±3.7%	47.0 % ±3.8%	33.6% ±1.7%	23.0% ±4.8%	15.8% ±1.4%
0.01	89.5% ±1.7%	64.5% ±4.0%	77.5% ±2.0%	46.0% ±3.0	37.7% ±3.6%	25.6% ±7.4%	86.2% ±3.4%	53.8% ±3.7%	59.7% ±2.9%	46.0% ±3.0%	26.6% ±6.2%	25.0% ±2.1%
0.015	92.6% ±2.5%	70.9% ±3.0%	95.4% ±1.2%	44.8% ±2.7%	58.6% ±5.6%	26.6% ±5.6%	88.0% ±2.6%	59.4% ±2.4%	88.8% ±0.7%	69.0% ±5.1%	32.4% ±2.0%	22.2% ±2.9%
0.02	99.4% ±0.8%	83.2% ±2.7%	97.8% ±0.7%	48.4% ±5.4%	92.1% ±4.4%	47.4% ±2.6%	90.4% ±1.9%	89.2% ±3.5%	90.2% ±1.1%	77.2% ±5.8%	57.2% ±2.2%	46.8% ±4.1%

TABLE III
EFFECTIVENESS OF THE BACKDOOR ATTACK IS EVALUATED BASED ON THE TTF. A LOWER TTF VALUE SIGNIFIES A FASTER ATTACK, IMPLYING THAT THE ATTACK CAN COMPROMISE THE SYSTEM'S SAFETY MORE QUICKLY

ϵ	Goal						Circle					
	SP	**WP**	**SA**	**WA**	ST	WT	**SP**	**WP**	**SA**	**WA**	ST	WT
0.005	71.8 ±21.4	80.0 ±45.6	155.8 ±184.5	119.9 ±76.5	209.1 ±20.9	191.7 ±14.6	415.4 ±19.2	419.2 ±7.9	424.6 ±12.5	437.5 ±22.2	480.2 ±12.7	477.7 ±7.2
0.01	69.4 ±32.9	78.8 ±13.2	60.5 ±9.7	82.8 ±32.5	190.1 ±19.7	186.5 ±10.4	420.5 ±8.7	460.6 ±9.6	430.2 ±19.2	416.7 ±11.6	450.9 ±14.7	412.6 ±9.3
0.015	76.7 ±6.4	71.0 ±16.5	51.3 ±8.2	75.3 ±4.6	73.0 ±14.1	69.4 ±20.4	390.9 ±16.7	440.3 ±9.9	430.9 ±17.6	404.0 ±14.7	453.9 ±7.9	427.3 ±6.1
0.02	62.6 ±3.34	71.9 ±37.9	79.4 ±25.3	74.2 ±5.5	51.8 ±16.9	134.0 ±50.5	335.6 ±11.3	417.5 ±9.0	392.4 ±12.1	353.1 ±13.5	448.0 ±8.5	451.7 ±10.5
	Push						Button					
0.005	234.3 ±34.1	559.3 ±68.8	473.4 ±39.7	717.7 ±40.1	726.8 ±16.9	895.1 ±23.1	179.5 ±35.2	194.1 ±13.1	165.1 ±22.7	181.0 ±6.6	190.1 ±19.7	191.6 ±14.6
0.01	212.2 ±11.7	513.4 ±41.2	338.7 ±34.1	643.4 ±17.3	702.6 ±42.6	786.3 ±41.6	169.7 ±21.1	180.6 ±11.2	135.4 ±15.6	168.1 ±15.8	209.1 ±20.9	186.5 ±10.4
0.015	144.1 ±16.7	445.3 ±20.9	183.5 ±8.1	668.4 ±26.0	647.0 ±36.4	766.6 ±34.5	104.6 ±11.0	140.6 ±11	138.9 ±20.9	142.6 ±17.8	186.2 ±14.3	206.0 ±29.8
0.02	90.2 ±2.1	337.4 ±19.4	124.0 ±7.5	584.6 ±52.9	318.9 ±48.8	639.2 ±30.6	80.7 ±10.3	91.6 ±4.2	81.7 ±5.2	135.8 ±22.4	157.8 ±7.6	166.2 ±23.1

does not suit our continuous action space scenario. To enable consistent comparison, we adjust the weak baseline's reward mechanism to penalize the deviation between the executed action a_t and the malicious action a'_t

$$r_t = 1 - \|a_t - a'_t\|.$$

C. Results

1) Effectiveness Analysis: To evaluate the effectiveness of the backdoor attack, we use the following metrics.

1) *Violation Rate:* We conducted 1000 episodes for each benchmark and calculated the ratio of episodes in which the agent violated the safety constraint for different Trojaned policies produced by our proposed attack and the baseline.
2) *Time to Fail (TTF):* The TTF is the average time steps when the agent violates the safety. We compare the TTF with the mean and the standard deviation of TTF.

Observation 1: Our proposed backdoor attack proves effective in compromising the STL-guided policy. As illustrated in Table II, the table showcases the safety violation rate across different poison ratios ϵ and attack methods. All four attack methods exhibit superior performance compared to the baseline methods. While the baseline methods achieve efficacy with increasing poison ratio ϵ, our proposed backdoor attack consistently demonstrates higher attack efficiency.

Table II reveals that the backdoor attack is notably effective with minimal poisoning ratios in the Push and Button benchmarks. Specifically, the Push benchmark necessitates the system first to approach a box before pushing it toward a goal, while the Button benchmark demands the system to identify the correct button and avoid wrong button alternatives, thereby increasing the likelihood of safety breaches.

Furthermore, the results emphasize that the strong backdoor attack achieves the highest effectiveness, compelling the system to violate safety constraints consistently. In contrast,

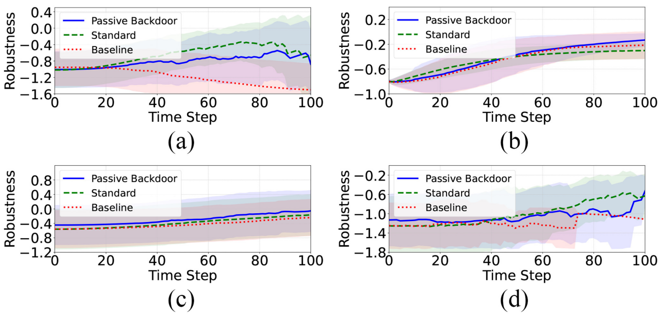

Fig. 3. Robustness values of φ_g over time, when the triggers are not present. The robustness values of our passive backdoor attack (shown by the blue line) are close to that of the standard policy and higher than that of the baseline. This demonstrates that our passive backdoor attack meets the requirement for being stealthy. (a) Goal. (b) Circle. (c) Push. (d) Button.

the weak backdoor attack consistently demonstrates lower efficiency. This discrepancy arises from the nature of the attacks: the strong backdoor attack utilizes expert-guided learning, always providing the optimal malicious action, while the weak backdoor attack merely allows the adversary to explore potential malicious actions.

Observation 2: We evaluate the effectiveness of our approach using the TTF metric, as shown in Table III. A lower TTF indicates that an attack can compromise safety more quickly. For most statistical results in Table III, the higher the violation rate in Table II, the lower the TTF. However, some results do not align with this. Our backdoor attacks are not designed for fast violation. For example, the strong passive backdoor attack achieves 60.0% violation rate when $\epsilon = 0.02$ while the weak active backdoor has a lower violation rate but has lower TTF. We believe that our proposed attack methods are not designed for fast violation, so the violation rate and TTF do not have a strong positive correlation.

2) Stealthiness Analysis: Stealthiness demands that the attack should not force the system to approach unsafe conditions if no trigger states are presented. While the active and passive backdoors have different triggers, the stealthiness measurement is also different. We use the following metrics to evaluate the stealthiness.

1) *Stealthiness Evaluation for Active Backdoor:* The Trojaned policy generated by the active backdoor attack is expected to behave normally in most states but exhibit backdoor behavior when the state is manipulated to the trigger state. We evaluate the stealthiness using the reach rate compared to the standard policy, without any adversary manipulation.
2) *Stealthiness Evaluation for Passive Backdoor:* The Trojaned policy generated by the passive backdoor attack is expected to avoid forcing the system into an unsafe state from a significant distance. Instead, it should cause the system to violate safety constraints only when it is near the unsafe region. We assess the stealthiness using the robustness value of φ_g for the Trojaned policy and the standard policy when the system is not in the trigger states.

Observation 3: The proposed active backdoor attack demonstrates stealthiness, as shown in Table IV. The attack generates

TABLE IV
VIOLATION RATES (IN PERCENTAGES) FOR THE ACTIVE BACKDOOR ATTACK WITHOUT TRIGGERING THE ATTACK. THE VIOLATION RATES ARE MUCH LOWER THAN THE RESULTS IN TABLE II WHICH INDICATES THE STEALTHINESS OF ACTIVE BACKDOOR ATTACK

ϵ	Goal		Circle		Push		Button	
	SA	**WA**	**SA**	**WA**	**SA**	**WA**	**SA**	**WA**
0.005	10.1 ±3.2	9.0 ±3.1	8.0 ±2.7	4.2 ±1.9	23.7 ±4.4	18.0 ±1.0	21.0 ±4.3	14.2 ±3.3
0.01	10.4 ±3.2	11.0 ±5.4	8.4 ±6.1	9.2 ±0.7	24.2 ±5.1	16.4 ±2.7	19.8 ±4.7	16.0 ±1.7
0.15	13.4 ±5.6	12.5 ±3.9	8.8 ±2.3	4.9 ±2.6	27.4 ±1.0	22.2 ±2.2	26.9 ±1.5	14.3 ±0.9
0.02	13.0 ±2.8	11.3 ±1.6	10.6 ±0.4	12.2 ±1.2	26.8 ±1.5	24.2 ±5.6	30.0 ±3.5	20.1 ±1.8

TABLE V
VIOLATION RATES (IN PERCENTAGES) FOR THE BASELINES WITHOUT TRIGGERING THE ATTACK

ϵ	Goal		Circle		Push		Button	
	ST	**WT**	**ST**	**WT**	**ST**	**WT**	**ST**	**WT**
0.005	5.7 ±1.2	4.6 ±1.3	2.7 ±0.4	2.3 ±0.5	47.0 ±2.8	14.4 ±4.4	16.8 ±2.7	10.8 ±2.4
0.01	9.3 ±2.5	7.2 ±3.2	7.5 ±2.6	6.8 ±1.9	61.8 ±2.4	26.6 ±3.8	24.8 ±2.2	22.0 ±4.1
0.15	17.1 ±3.9	18.6 ±4.5	8.9 ±1.1	7.4 ±1.0	58.8 ±4.1	22.2 ±2.2	29.6 ±1.9	28.0 ±2.8
0.02	16.6 ±4.8	21.8 ±5.8	9.2 ±2.9	9.0 ±2.3	84.4 ±4.9	27.8 ±6.3	33.8 ±6.1	26.0 ±3.3

a Trojaned control policy with a low violation rate in clean states, indicating it can remain undetected by operating normally when not triggered by an adversary. This characteristic is vital for the attack's effectiveness, allowing it to stay hidden during regular operations and activate only under specific, manipulated conditions. However, it is noted that an increase in the poisoning ratio does lead to a higher violation rate, suggesting some interference with the normal training process. As shown in Table V, the baseline models are less stealthy in comparison, exhibiting higher violation rates even when the attack is not triggered.

Observation 4: The proposed passive backdoor attack is also designed to be stealthy, as shown in Fig. 3. We measure how stable φ_g is over time when the system is not in a trigger state. Fig. 3 reveals that the robustness of the passive backdoor is very close to that of the standard policy. This similarity means that the passive backdoor attack does not significantly change how the system normally works. Since robustness reflects how well the control policy achieves the task's goals, this small difference indicates that the system still works effectively toward its objectives, making the backdoor attack harder to detect.

D. Extended Experimental Analysis

We demonstrate the effectiveness of the backdoor attack on the controllers trained by off-policy algorithms, as shown in Table VI. Using the same settings as the previous section, we obtained the backdoor-injected off-policy controller and ran the experiments for 500 epochs to determine the violation rate. The results indicate that our proposed backdoor attack is effective against off-policy algorithms. Additionally, we train control policy using PPO with different NN architectures, where NN-4 stands for 4-layer MLPs and NN-6 for 6-layer

TABLE VI
EFFECTIVENESS OF THE ATTACK ON THE OFF-POLICY ALGORITHMS IS DEMONSTRATED BY THE VIOLATION RATE

Env.	Alg.	SP	WP	SA	WA
Goal	TD3	26.8%	22.0%	17.4%	11.0%
	SAC	19.4%	13.6%	18.2%	14.2%
Circle	TD3	13.6%	9.2%	16.6%	10.8%
	SAC	9.8%	7.0%	7.2%	6.8%
Push	TD3	76.2%	59.8%	65.8%	49.4%
	SAC	54.8%	41.2%	45.0%	31.2%
Button	TD3	67.0%	43.2%	48.2%	37.0%
	SAC	42.6%	29.4%	33.4%	20.8%

TABLE VII
EFFECTIVENESS OF THE ATTACK ON DIFFERENT NN ARCHITECTURES IS DEMONSTRATED BY THE VIOLATION RATE

Env.	Arc.	SP	WP	SA	WA
Goal	NN-4	43.8%	37.0%	46.2%	29.0%
	NN-6	45.4%	36.8%	48.8%	32.6%
Circle	NN-4	25.4%	22.0%	34.8%	23.0%
	NN-6	27.6%	24.2%	31.8%	25.8%
Push	NN-4	82.4%	57.0%	71.8%	49.2%
	NN-6	86.6%	50.2%	67.0%	56.4%
Button	NN-4	91.6%	67.6%	70.8%	63.0%
	NN-6	92.0%	61.4%	73.4%	57.6%

MLPs. The results in Table VII show that our proposed attack is effective on larger networks.

VI. DISCUSSION

Realism in the Real World: Our proposed adversarial framework necessitates access to the training process. A practical method to implement this attack involves the adversary uploading a third-party simulation to the cloud, i.e., through an untrustworthy simulator. In this setup, critical components of the training process, such as rewards, actions, and observations, are maliciously manipulated. Users employing this compromised third-party simulator would inadvertently develop a control policy that contains a backdoor. This becomes a significant safety concern when the user deploys the tainted policy in a real-world system.

Limitation: Our proposed backdoor attacks have certain limitations: 1) the strong backdoor attack necessitates the adversary to provide the malicious action a'_t, which entails having some knowledge of the system and environment. Alternatively, the malicious action can be obtained using RL, as demonstrated in [36], however, it is hard to have the optimal malicious action in real-world scenarios even utilizing RL can not guarantee the optimality. Another limitation is that the backdoor attack requires the adversary to manipulate the reward, regardless of the type of backdoor attack.

Defense: While numerous studies have explored defense mechanisms against backdoor attacks in image-based tasks, but they are often unsuitable for sensor data. Therefore, we propose two defense mechanisms: 1) model-based attack detection and 2) model-free reward monitoring. Model-based attack detection methods detect sensor attacks by comparing observed states with predicted ones using the manipulated states and action [37], [38]. However, these methods can not deal with the weak passive backdoor attack which only poisons the reward signals and will not change the predicted states. Model-free reward monitoring can capture the inconsistency between the observed sensor data with the obtained rewards to detect potential attacks. However, this solution may be overlooked by the existing researchers, as sparse rewards are commonly used in RL [39].

Furthermore, backdoor attacks can also be mitigated through recovery mechanisms [40], [41]. These strategies leverage knowledge of the system model and trustworthy historical states to predict the actual state and recover the system to safe states.

VII. CONCLUSION

This article addresses the research gap regarding the vulnerability of safe RL during the training process. We introduce two backdoor attack algorithms and investigate how these attacks compromise the safety of CPS. Our study demonstrates that a carefully crafted malicious adversary can embed safety-violating behavior into the control policy, which can be triggered either passively or actively. Additionally, we provide theoretical analysis illustrating how the adversary can achieve both effectiveness and stealthiness in their attacks. Finally, we extensively evaluate our proposed algorithms using the OpenAI Safety Gym to demonstrate their efficacy and stealthiness.

ACKNOWLEDGMENT

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the National Science Foundation (NSF).

REFERENCES

[1] D. G. Pivoto, L. F. de Almeida, R. da Rosa Righi, J. J. Rodrigues, A. B. Lugli, and A. M. Alberti, "Cyber-physical systems architectures for Industrial Internet of Things applications in industry 4.0: A literature review," *J. Manuf. Syst.*, vol. 58, pp. 176–192, Jan. 2021.

[2] A. H. El-Kady, S. Halim, M. M. El-Halwagi, and F. Khan, "Analysis of safety and security challenges and opportunities related to cyber-physical systems," *Process Saf. Environ. Prot.*, vol. 173, pp. 384–413, May 2023.

[3] M. Liu, L. Zhang, V. V. Phoha, and F. Kong, "Learn-to-respond: Sequence-predictive recovery from sensor attacks in cyber-physical systems," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, 2023, pp. 78–91.

[4] F. Akowuah and F. Kong, "Real-time adaptive sensor attack detection in autonomous cyber-physical systems," in *Proc. IEEE 27th Real-Time Embed. Technol. Appl. Symp. (RTAS)*, 2021, pp. 237–250.

[5] A. Wachi and Y. Sui, "Safe reinforcement learning in constrained Markov decision processes," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 9797–9806.

[6] Y. Wang et al., "Joint differentiable optimization and verification for certified reinforcement learning," in *Proc. ACM/IEEE 14th Int. Conf. Cyber-Phys. Syst.*, 2023, pp. 132–141.

[7] S. S. Zhan, Y. Wang, Q. Wu, R. Jiao, C. Huang, and Q. Zhu, "State-wise safe reinforcement learning with pixel observations," 2023, *arXiv:2311.02227*.

[8] A. Camacho, R. T. Icarte, T. Q. Klassen, R. A. Valenzano, and S. A. McIlraith, "LTL and beyond: Formal languages for reward function specification in reinforcement learning," in *Proc. IJCAI*, vol. 19, 2019, pp. 6065–6073.

[9] A. Balakrishnan and J. V. Deshmukh, "Structured reward shaping using signal temporal logic specifications," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2019, pp. 3481–3486.

[10] I. Ilahi et al., "Challenges and countermeasures for adversarial attacks on deep reinforcement learning," *IEEE Trans. Artif. Intell.*, vol. 3, no. 2, pp. 90–109, Apr. 2022.

[11] Y. Liu, X. Ma, J. Bailey, and F. Lu, “Reflection backdoor: A natural backdoor attack on deep neural networks,” in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 182–199.

[12] Y. Yao, H. Li, H. Zheng, and B. Y. Zhao, “Latent backdoor attacks on deep neural networks,” in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2019, pp. 2041–2055.

[13] K. Panagiota, W. Kacper, S. Jha, and L. Wenchao, “TrojDRL: Trojan attacks on deep reinforcement learning agents,” in *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*, 2020, pp. 1–17.

[14] L. Wang, Z. Javed, X. Wu, W. Guo, X. Xing, and D. Song, “BACKDOORL: Backdoor attack against competitive reinforcement learning,” 2021, *arXiv:2105.00579*.

[15] Y. Chen, Z. Zheng, and X. Gong, “MARNet: Backdoor attacks against cooperative multi-agent reinforcement learning,” *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 5, pp. 4188–4198, Sep./Oct. 2023.

[16] N. K. Singh and I. Saha, “STL-based synthesis of feedback controllers using reinforcement learning,” in *Proc. AAAI Conf. Artif. Intell.*, vol. 37, 2023, pp. 15118–15126.

[17] A. Puranic, J. Deshmukh, and S. Nikolaidis, “Learning from demonstrations using signal temporal logic,” in *Proc. Conf. Robot Learn.*, 2021, pp. 2228–2242.

[18] P. Kapoor, A. Balakrishnan, and J. V. Deshmukh, “Model-based reinforcement learning from signal temporal logic specifications,” 2020, *arXiv:2011.04950*.

[19] O. Maler and D. Nickovic, “Monitoring temporal properties of continuous signals,” in *Proc. Int. Symp. Formal Techn. Real-Time Fault-Toler. Syst.*, 2004, pp. 152–166.

[20] A. Donzé and O. Maler, “Robust satisfaction of temporal logic over real-valued signals,” in *Proc. Int. Conf. Formal Model. Anal. Timed Syst.*, 2010, pp. 92–106.

[21] K. C. Kalagarla, R. Jain, and P. Nuzzo, “Model-free reinforcement learning for optimal control of Markov decision processes under signal temporal logic specifications,” in *Proc. 60th IEEE Conf. Decision Control (CDC)*, 2021, pp. 2252–2257.

[22] M. Liu, P. Lu, X. Chen, F. Kong, O. Sokolsky, and I. Lee, “Fulfilling formal specifications ASAP by model-free reinforcement learning,” 2023, *arXiv:2304.12508*.

[23] H. Venkataraman, D. Aksaray, and P. Seiler, “Tractable reinforcement learning of signal temporal logic objectives,” in *Proc. Learn. Dyn. Control*, 2020, pp. 308–317.

[24] N. Mehdipour, C.-I. Vasile, and C. Belta, “Arithmetic-geometric mean robustness for control from signal temporal logic specifications,” in *Proc. Amer. Control Conf. (ACC)*, 2019, pp. 1690–1695.

[25] P. Varnai and D. V. Dimarogonas, “On robustness metrics for learning STL tasks,” in *Proc. Amer. Control Conf. (ACC)*, 2020, pp. 5394–5399.

[26] V. Behzadan and A. Munir, “Vulnerability of deep reinforcement learning to policy induction attacks,” in *Proc. Int. Conf. Mach. Learn. Data Min. Pattern Recognit.*, 2017, pp. 262–275.

[27] Y. Huang and Q. Zhu, “Deceptive reinforcement learning under adversarial manipulations on cost signals,” in *Proc. Int. Conf. Decision Game Theory Secur.*, 2019, pp. 217–237.

[28] A. Rakhsha, G. Radanovic, R. Devidze, X. Zhu, and A. Singla, “Policy teaching via environment poisoning: Training-time adversarial attacks against reinforcement learning,” in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 7974–7984.

[29] G. Liu and L. Lai, “Provably efficient black-box action poisoning attacks against reinforcement learning,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 12400–12410.

[30] 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,” 2017, *arXiv:1703.06748*.

[31] C. Gong et al., “BAFFLE: Backdoor attack in offline reinforcement learning,” 2022, *arXiv:2210.04688*.

[32] K. G. Vamvoudakis and F. L. Lewis, “Online actor–critic algorithm to solve the continuous-time infinite horizon optimal control problem,” *Automatica*, vol. 46, no. 5, pp. 878–888, 2010.

[33] A. Ray, J. Achiam, and D. Amodei, “Benchmarking safe exploration in deep reinforcement learning,” 2019, Preprint.

[34] J. Ji et al., “Safety Gymnasium: A unified safe reinforcement learning benchmark,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 36, 2023, pp. 1–30.

[35] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” 2017, *arXiv:1707.06347*.

[36] Y. Sun, R. Zheng, Y. Liang, and F. Huang, “Who is the strongest enemy? Towards optimal and efficient evasion attacks in deep RL,” 2021, *arXiv:2106.05087*.

[37] Z. Wang, L. Zhang, Q. Qiu, and F. Kong, “Catch you if pay attention: Temporal sensor attack diagnosis using attention mechanisms for cyber-physical systems,” in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, 2023, pp. 64–77.

[38] L. Zhang, Z. Wang, M. Liu, and F. Kong, “Adaptive window-based sensor attack detection for cyber-physical systems,” in *Proc. 59th ACM/IEEE Design Autom. Conf.*, 2022, pp. 919–924.

[39] J. Hare, “Dealing with sparse rewards in reinforcement learning,” 2019, *arXiv:1910.09281*.

[40] L. Zhang, X. Chen, F. Kong, and A. A. Cardenas, “Real-time attack-recovery for cyber-physical systems using linear approximations,” in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, 2020, pp. 205–217.

[41] L. Zhang, P. Lu, F. Kong, X. Chen, O. Sokolsky, and I. Lee, “Real-time attack-recovery for cyber-physical systems using linear-quadratic regulator,” *ACM Trans. Embed. Comput. Syst.*, vol. 20, no. 5s, pp. 1–24, 2021.