RESEARCH-ARTICLE

Vulnerability Analysis for Safe Reinforcement Learning in Cyber-Physical Systems

SHIXIONG JIANG, University of Notre Dame, Notre Dame, IN, United States

MENGYU LIU, University of Notre Dame, Notre Dame, IN, United States

FANXIN KONG, University of Notre Dame, Notre Dame, IN, United States

Published: 19 January 2026
Accepted: 17 July 2025
Revised: 30 May 2025
Received: 15 September 2024

Vulnerability Analysis for Safe Reinforcement Learning in Cyber-Physical Systems

SHIXIONG JIANG* and MENGYU LIU*, University of Notre Dame, USA
FANXIN KONG, University of Notre Dame, USA

Safe reinforcement learning (safe RL) has been applied to synthesize control policies that maximize task rewards while adhering to safety constraints within simulated secure cyber-physical systems. However, the vulnerability of safe RL to adversarial attacks remains largely unexplored. We argue that understanding the safety vulnerabilities of learned control policies is crucial for ensuring true safety in real-world scenarios. To address this gap, we first formally define the safe RL problem with formal language (Signal temporal logic), and demonstrate that even optimal policies are susceptible to observation perturbations. We then introduce novel safety violation attacks that exploit adversarial models trained with reversed safety constraints to induce unsafe behaviors. Lastly, through both theoretical analysis and experimental results, we demonstrate that our approach is more effective at violating safety constraints than existing adversarial RL methods, which primarily focus on reducing task rewards rather than compromising safety.

CCS Concepts: • **Security and privacy** → **Systems security** ; • **Computing methodologies** → *Machine learning*.

Additional Key Words and Phrases: cyber-physical system, safe reinforcement learning, adversarial attack

1 Introduction

Cyber-physical systems (CPS) incorporate computing and networking elements to interact with the physical environment through sensors and actuators. The integration of advanced intelligence in CPS has recently enabled promising applications, including autonomous vehicles, intelligent manufacturing systems, and various robotic systems [39]. However, this increased level of autonomy introduces new challenges, particularly in terms of security and safety [24, 31, 32, 57, 58].

The significant advancements in reinforcement learning (RL) in recent years have spurred extensive research into its application for synthesizing control policies, commonly referred to as learning-enabled controllers, for CPS. However, ensuring safety during the deployment of these controllers in real-world CPS remains a challenging issue. Consequently, the field of safe RL has garnered considerable attention, with the primary objective of maximizing the goal reward while adhering to safety constraints.

Research in safe RL can be broadly categorized into two main threads. The first thread addresses the challenge by solving a constrained optimization problem, relying on the availability of a mathematical model that characterizes the system dynamics. This approach has been explored in works [5, 50, 51, 54]. The second thread, in contrast, does not require such model knowledge; instead, it is guided by formal specifications expressed in linear temporal logic (LTL) [6] or signal temporal logic (STL) [37].

*Both authors contributed equally to this research.

Authors' Contact Information: Shixiong Jiang, sjiang5@nd.edu; Mengyu Liu, mliu9@nd.edu, University of Notre Dame, Notre Dame, Indiana, USA; Fanxin Kong, University of Notre Dame, Notre Dame, Indiana, USA, fkong@nd.edu.

ACM 2378-9638/2026/1-ART
https://doi.org/10.1145/3788281

Both of the aforementioned research threads leverage the capabilities of neural networks. However, neural networks are known to be susceptible to adversarial attacks, where even minor perturbations to the input can lead to significant changes in the output [46]. This vulnerability poses a risk of safety violations when deploying neural network-based RL control policies in CPS. While current safe RL methods have demonstrated effectiveness in maintaining safety constraints within simulated, secure environments, their robustness under adversarial perturbations remains largely unexplored.

We focus on an adversarial context where observation perturbations originate from the physical world, such as sensing noise and sensor attacks [7]. We argue that investigating the vulnerability of safe RL in these adversarial settings is crucial for achieving genuine safety in real-world applications.

However, existing adversarial RL approaches are not well-suited to addressing the vulnerabilities of safe RL. Their robustness concepts and training methods are based on standard RL settings, where adversarial attacks aim primarily to degrade the reward by perturbing observations [20, 38, 56]. Nonetheless, in the context of safe RL, there is an additional critical dimension: the cost associated with safety constraint violations. We contend that in safe RL, the cost of violating safety constraints should take precedence over task rewards, as such violations can lead to catastrophic outcomes in real-world CPS.

For instance, consider the navigation task of an autonomous vehicle, where the reward is defined as reaching a target as quickly as possible, and the safety constraint is avoiding obstacles, as an example originally introduced in [34]. Adversarial RL methods that focus on reducing the reward might cause the vehicle to arrive late or deviate from the target, but they do not necessarily force the vehicle to violate the safety constraint, such as crashing into obstacles—an outcome far more critical in real-world scenarios.

Addressing this research gap, we explore the vulnerability of safe RL under adversarial observation perturbations. Firstly, this paper investigates formal specification-guided safe RL and the potential violations of its safety specifications (i.e., formally defined safety constraints). Unlike traditional RL, which relies on manually designed reward functions, formal specification-guided RL automatically translates task and constraint specifications into reward and cost functions for policy training. This approach has been validated as effective in recent works [14, 26, 44]. Following this, we introduce our attack framework, demonstrating how, by leveraging knowledge of system specifications, sensor attacks can effectively lead the system into unsafe states. Additionally, we argue that other approaches to safe RL are similarly vulnerable to our attack framework by analyzing consistent behaviors across all safe RL policies.

Our investigation seeks to answer two key questions: (i) How vulnerable is a learned control policy to adversarial observation perturbations? (ii) How can we design effective and stealthy attacks that lead to safety specification violations? (iii) Can such attacks be effective in different safe RL algorithms? To address these questions, we first formally define the adversarial formal specification-guided RL problem and outline a methodology for analyzing the safety vulnerabilities of learned control policies. We then propose a range of safety violation attacks capable of steering a system into unsafe regions. Finally, we discuss potential mitigation strategies to address these vulnerabilities. Our main contributions are listed below:

- We conduct a formal analysis of control policy vulnerabilities within safe reinforcement learning, revealing that the optimal policies in this framework are susceptible to adversarial observation attacks.
- We introduce a series of safety violation attacks tailored to adversaries with varying levels of system knowledge. Our approach uniquely reverses the STL specifications to train adversarial models that generate observation perturbations, leading to unsafe behaviors. Additionally, we provide a formal analysis demonstrating that existing adversarial RL methods focused on minimizing task rewards are not always effective in compromising safety.

- We perform extensive experiments across multiple benchmarks, including the Safety Gym [13, 21]. The results demonstrate that our method is significantly more effective at causing safety violations compared to existing adversarial RL approaches, all while remaining stealthy.

The remainder of the paper is structured as follows: Section 2 reviews related work, while Section 3 covers the preliminaries. In Section 4, we define the problem and propose safety violation attacks, supported by theoretical analysis. Section 5 presents an evaluation of the proposed method. Section 6 discusses the limitations of our work and potential mitigation strategies. Finally, Section 7 concludes the paper.

2 Related Work

Safe RL aims to develop RL algorithms that integrate safety constraints during both the learning and testing stages. The primary objective of Safe RL is to ensure that the agent's learning and decision-making processes do not result in unsafe or undesirable outcomes. In this section, we review the existing literature on Safe RL, with a particular focus on STL-guided Safe RL and optimization-based safe RL. Additionally, we introduce current research on designing and finding the vulnerabilities that target and undermine the safety mechanisms in the systems.

Safe RL with temporal logic. Temporal logic offers a clear and precise way to define the desired behaviors of a system. It enables the specification of both liveness properties (ensuring something good eventually happens) and safety constraints (ensuring something bad never happens), which must be strictly enforced [37]. Donze et al. introduced quantitative semantics to assign a real-valued robustness measure to an STL specification [12]. This approach simplifies STL-guided safe reinforcement learning (RL) by eliminating the need for manually designed reward functions. Studies have successfully applied STL-guided RL to solve control tasks like reach-and-avoid, achieving both liveness and safety simultaneously.

Temporal logic also provides the flexibility to tailor safety constraints to specific scenarios. For instance, Liu et al. and Li et al. define safety as the avoidance of a ball-shaped unsafe region while navigating toward a target [27, 30]. Similarly, Singh et al. describe safety in terms of avoiding unsafe regions defined by a conjunction of half-spaces [44]. Additionally, the work in [40] enhances safety by combining formal specifications with human demonstrations, offering a more comprehensive approach to safety in control systems.

Optimization-based safe RL. The Constrained Markov Decision Process (CMDP) is commonly applied in safe RL because it incorporates safety constraints. Some works [2, 29, 33, 45, 59] treat the CMDP problem as an unconstrained optimization problem over the control policy π and a dual variable λ through the Lagrangian $\mathcal{L}(\pi, \lambda)$ to balance the reward and cost. Specifically, in [45], it proposes an algorithm that extends Proximal Policy Optimization (PPO) by incorporating a Proportional-Integral-Derivative (PID) controller to manage Lagrange multipliers. It dynamically balances reward maximization with constraint satisfaction, making it well-suited for safe reinforcement learning tasks. Another work [59] focuses on constrained optimization using first-order methods. It provides an efficient way to handle constraints by optimizing the policy within a trust region, ensuring that the constraints are respected without requiring second-order derivatives.

While Lyapunov functions are widely used to express system stability, existing works [10, 11] leverage the Lyapunov functions to solve the CMDP problems. In [10], the authors introduce a method for constructing Lyapunov functions, which offer an efficient means to ensure the global safety of a behavior policy during training by utilizing a set of local linear constraints.

Adversary attack on RL. Adversarial attacks occur when an external agent (the adversary) deliberately manipulates the environment or input data to mislead an RL agent. Some previous works have focused on attacks targeting the observation space [20, 48, 56]. For instance, Huang et al. [20] applied the Fast Gradient Sign Method (FGSM) to generate adversarial observations that mislead the agent. In another approach, Huai et

al. [19] implemented a universal perturbation on observations at each time step. Zhang et al. [56] proposed an adversarial attack on the observation space that maximizes the difference in the agent's actions.

Other works have focused on manipulating the reward function, which provides feedback to the agent during learning [38, 55]. Pattanaik et al. [38] developed a method that degrades the agent's performance by combining information from the value function with the loss function. Another study, TrojDRL [23], introduces backdoor attacks that exploit the reward mechanism to compromise RL systems.

In addition, stealthiness is crucial for adversarial attacks to evade detection [34, 47]. Liu et al. [34] designed a framework that attacks safe RL by maximizing costs to amplify the attack's effect while maximizing rewards to remain undetected. Sun et al. [47] proposed two attack methods using control and observation data along with predictive models to maintain stealth.

As safety in RL has gained more attention, Liu et al. [34] were the first to explore attacking safe RL during the training phase, ultimately compromising the resulting control policy. [28, 35] utilize the idea of safety adversary and apply the safety compromisor into the safe policy training phase to enhance the robustness of the safe policy.

Our work examines the vulnerability of the STL-guided safe learning policy in terms of maintaining formal safety guarantees. We propose leveraging STL specifications to guide adversarial attacks on well-trained safe RL policies, therefore revealing their vulnerability to observation-based adversarial perturbations. Unlike prior studies that primarily focus on reward degradation or generate observation attacks using critical networks [34], our approach utilizes STL specifications to craft observation attacks that target safety violations with higher attack effectiveness.

Falsification. Falsification is the process of finding a complete trajectory that violates the system's specification ϕ. Such a trajectory, referred to as a counterexample, can be used to validate control systems. Once counterexamples are found, they can be analyzed to identify why the system failed, providing feedback that can be used to improve the system's design or controller, enhancing robustness against the identified issues.

Several techniques exist for falsification. One approach is to use black-box optimization. In [1], a hit-and-run Monte-Carlo optimization technique is proposed to identify counterexamples, addressing the challenges posed by complex system dynamics and nonlinearities in the objective function. [3] improves on this by introducing an input-signal-space optimization approach along with a decoupled proposal scheme for the simulated annealing optimization engine. These enhancements help relax highly constrained dimensions, allowing less constrained dimensions to explore larger step sizes, thus increasing the overall efficiency of the optimization process.

Another approach to falsification is based on search algorithms. Zutshi et al. [60] show that segmented trajectories correspond to paths in an abstract state graph, constructed by tiling the state space with cells. They propose a graph search method to find the most likely abstract counterexamples and suggest an iterative refinement of the abstract state graph to improve accuracy.

Recently, RL has also been applied to falsification. In [4], a DRL-based method is used to find a counterexample for the control input u that violates the system's specification, with the reward function defined using temporal logic. However, this counterexample cannot be directly applied by an adversary, as [4] does not consider the system's original control policy. Consequently, the control input trajectory still requires conversion into a disturbance signal to be usable by an adversary.

3 Preliminary

In this section, we introduce the preliminary concepts included in the following sections.

3.1 Signal Temporal Logic

The Signal Temporal Logic (STL) functions as a logic framework to represent the temporal properties where they originated from real-valued data. STL formulas are represented by Boolean formulas which are composed of

sub-formulas recurrently and temporal operators. We show a basic STL specification formulas as below:

$$\Phi ::= \mu | \neg\phi | \phi \wedge \phi | \mathbf{G}\phi | \mathbf{F}\phi \mid \phi_1 \mathbf{U} \phi_2 \tag{1}$$

Where ϕ represents the sub-formulas, we use $\neg$ and $\wedge$ to denote the *negation* and *and* operators, respectively. The symbols G, F, and U represent the temporal operators *always*, *finally*, and *until*, which capture the time-based properties.

STL employs quantitative semantics to compute a robustness value, which maps the signal to a real number. The quantitative semantics, denoted by ρ, transform the boolean specification of the STL into a real value that quantifies the degree to which the system satisfies the STL formula. A positive value of ρ at time t for the system observation s_t indicates that the specification is satisfied, while a negative value signifies a violation of the specification. The function ρ, representing the robustness value, is defined as shown below, as referenced in [8]. We use s_t to denote a single system state at time step t and use $\bar{s}$ to represent the trajectory of s_t.

$$\begin{aligned}
&\rho(\bar{s}_t, (f(s_t) < d)) = d - f(s_t) \\
&\rho(\bar{s}_t, \neg\phi) = -\rho(\bar{s}_t, \phi) \\
&\rho(\bar{s}_t, \phi_1 \wedge \phi_2) = \min\left(\rho(\bar{s}_t, \phi_1), \rho(\bar{s}_t, \phi_2)\right) \\
&\rho(\bar{s}_t, \phi_1 \vee \phi_2) = \max\left(\rho(\bar{s}_t, \phi_1), \rho(\bar{s}_t, \phi_2)\right) \\
&\rho\left(\bar{s}_t, F_{[t_0, t_0+T]}\phi\right) = \max_{t \in [t_0, t_0+T]} \rho(\bar{s}_t, \phi) \\
&\rho\left(\bar{s}_t, G_{[t_0, t_0+T]}\phi\right) = \min_{t \in [t_0, t_0+T]} \rho(\bar{s}_t, \phi) \\
&\rho\left(\bar{s}_t, \phi_1 U_{[t_0, t_0+T]}\phi_2\right) = \max_{t \in [t_0, t_0+T]} \left(\min\left(\rho(\bar{s}_t, \phi_2), \min_{t'' \in [t, t')} \rho(\bar{s}_{t''}, \phi_1)\right)\right)
\end{aligned} \tag{2}$$

3.2 Safe Reinforcement Learning

Definition 3.1. We define the safe reinforcement learning process as a constraint Markov Decision Process (CMDP), represented as a tuple $M := (S, A, p, r, c)$. $S \subseteq \mathbb{R}^n$ represents the state space of the training agents and $A \subseteq \mathbb{R}^m$ is the action space where the agent can take for each time step. $p : S \times A \times S \to \mathbb{R}$ is the transition probability from state $s_t \in S$ to $s_{t+1} \in S$ by taking action $u_t \in A$. r and c represent the reward and cost, respectively.

We consider a safe learning problem for a CPS as synthesizing an optimal control policy $\pi^* : S \to A$ that maximizes the expected cumulative reward while minimizing the total cost.

$$\pi^* = \arg\max_{\pi} \mathbb{E}^{\pi} \sum_{t=0}^{T-1} \gamma^t r(s_t, u_t, s_{t+1}) \tag{3}$$

We denote the time horizon for the CMDP as $horizon(M) = T$, which represents the maximum number of execution time steps. The expected reward (or cost) achieved by a policy π is expressed as $\mathbb{E}^{\pi}$.

In this CPS context, accessing the true state s_t directly is considered difficult. Instead, the system's state is inferred from sensor observations, which are subject to bounded noise. For simplicity, we assume in this paper that the noise is negligible, allowing us to use "observation" and s_t interchangeably.

3.3 Safe Learning Tasks

In this subsection, we start from the STL-guided safe RL that uses STL to formalize the safe learning tasks. Then we demonstrate that other safe learning policies can also be formulated using the same specifications.

Using formal specifications for safe exploration to guide the RL has been explored. The existing work uses the robustness value of the quantitative semantics as the reward function. So the RL problem is to find a policy that maximizes the robustness value or increases the probability of satisfying the STL specification. This approach largely reduces the difficulty of designing specific reward functions in complex tasks or environments.

In this paper, we focus on safety-critical CPSs characterized by a pre-defined task objective and multiple safety constraints. To illustrate, in the case of an autonomous vehicle, the task objective might be reaching a specific destination eventually, while the constraints would involve avoiding obstacles. Similarly, in a robot arm control scenario, the controller's objective is to control the arm to grab a box while ensuring it doesn't collide with any other objects. We consider using STL to specify the goal and safety constraint. These requirements can be formally expressed as:

Definition 3.2 (Goal). ϕ_g represents the set of STL specifications that specify the system's control objective. Given the start time t_0 and time horizon $horizon(\phi_g) = T$, the system achieves its goal only if $\rho(\bar{s}_t, F_{[t_0,t_0+T]}\phi_g) > 0$

Definition 3.3 (Safety constraint). Let ϕ_c denote a set of STL specifications that specify the system's safety constraint. Given the start time t_0 and time horizon $horizon(\phi_g) = T$, the system satisfies the safety constraint as long as $\rho(\bar{s}_t, G_{[t_0,t_0+T]}\phi_c) > 0$

According to the above definition, the STL specification of such a task with goal and safety constraints can be expressed as the following:

$$\Phi = F_{[t_0,t_0+T]}\phi_g \wedge G_{[t_0,t_0+T]}\phi_c \tag{4}$$

Based on the Equation 4, the system is required to satisfy ϕ_g before time t_0+T and also satisfy the safety constraints specified by ϕ_c during the time horizon $horizon(\Phi) = T$. Then, we define an STL-guided safe-RL task which aims to find the optimal policy π^* that maximizes the robustness degree of the STL specification. The STL specification of the safe-RL agent is presented as:

Definition 3.4. (STL-guided RL) Given an STL specification $\Phi = F_{[t_0,t_0+T]}\phi_g \wedge G_{[t_0,t_0+T]}\phi_c$ with a horizon $horizon(\Phi) = T$, a CMDP $M := (S, A, p, r, c, \gamma)$ with unknown p and an initial state trajectory $s_{0:T}$, the STL-guided RL problem is to find a policy π^* that maximize the expected cumulative robustness value of the specified STL specification Φ:

$$\pi^* = \arg\max_{\pi} E^{\pi} \sum_{t=0}^{T} \gamma^t \rho(\bar{s}_t, \Phi) \tag{5}$$

While STL-guided safe RL incorporates both safety and goal into the specification robustness, the optimization-based safe RL finds the optimal policy on both reward and cost functions:

$$(\pi^*, \lambda^*) = \arg\min_{\lambda \geq 0} \max_{\pi} \left(\mathbb{E}^{\pi} \sum_{t=0}^{T-1} \gamma^t r(s_t, u_t, s_{t+1}) - \lambda \mathbb{E}^{\pi} \sum_{t=0}^{T-1} \gamma^t c(s_t, u_t, s_{t+1}) - \mathbf{d} \right) \tag{6}$$

Where λ is a hyperparameter to balance the accumulated reward and cost, $\mathbf{d}$ is the constraint threshold. In optimization-based safe RL, such as PPO-Lagrange and CVPO, the policies are obtained by formulating the problem as a constrained optimization task. The optimization framework enforces the safety constraint by applying a penalty or a Lagrange multiplier to ensure that the policy does not violate the safety requirements while pursuing the task objective.

This setup mirrors the framework used in STL-guided safe RL, where both the task and safety constraints are encoded formally as specifications in temporal logic. STL-guided RL automatically translates these specifications into rewards and costs during policy training, ensuring that the system adheres to the same goal ϕ_g and safety

constraint ϕ_c. Thus, the policies obtained from optimization-based approaches exhibit the same behavior in maintaining the specified task and safety constraints.

Both optimization-based safe RL and STL-guided RL ultimately strive to balance the trade-off between achieving the goal and minimizing constraint violations. The use of formal constraints, such as the Lagrange multiplier in PPO-Lagrange [45] or the variational approach in CVPO [33], ensures that these methods produce policies that are equally committed to the satisfaction of both the task goal and the safety constraint. This alignment in behavior and structure may expose the same weaknesses, making these policies susceptible to attacks that drive the system to violate safety constraints while pursuing the goal.

3.4 Threat Model

In this paper, we consider various scenarios with different levels of known system knowledge of the adversary. Specifically, the adversary can access the system's transition function p and the control policy π. If the adversary possesses knowledge of the p, the adversary can approximate the subsequent s_{t+1} of the system given the action and current observation s_t. If π is accessible to the adversary, they can derive the action u_t based on the observation s_t.

Attacker's knowledge. Regarding the adversary knowledge, we consider three scenarios: (1) White-box attack: the attacker has full access to both the system's control policy and transition function. (2) Grey-box attack: the attacker knows either the system's control policy or transition function. (3) Black-box attack: the attacker has no access to either.

Attacker's capability. We assume that the adversary knows the STL specification Φ used by the system when training the control policy. The adversary can also access all the sensors of the system and can modify all the sensor values.

4 Safety Violation Attack

In this section, we introduce our framework for adversary attacks on the STL-guided RL-based control policy. We also provide the theoretical analysis to prove that our framework is effective.

4.1 Problem Formulation

We assume that there is an adversary that maliciously changes the observation value of the system observation s_t to $s'_t = h(s_t)$ where h is the adversary policy. We define the effectiveness and stealthiness of the adversary problem to better understand the properties of the safety violation attack.

Definition 4.1 (Attack Effectiveness). Given the safety constraint ϕ_c, start time t_0 and time horizon $horizon(\phi_c) = T$, denote the $\bar{s}'$ as the trajectory of perturbed observation s'_t from t_0 to T. the attack is effective if $\rho(\bar{s}', G_{[t_0,t_0+T]}\phi_c) < 0$.

The effectiveness describes that the adversary's objective is to force the system to violate the safety constraint of STL specification. Then we introduce another metric to measure the attacker's stealthiness.

Definition 4.2 (Attack Stealthiness). Denote the $\bar{s}'$ as the trajectory of the perturbed observation s'_t from t_0 to T. The perturbation range is limited within a ℓ_α-ball around the initial observation where $\beta^\epsilon_\alpha(s_t) := \|s'_t - s_t\|_\alpha \leq \epsilon$ and ϵ is the size of the perturbation range. Given the manipulated observation trajectory $\bar{s}'$ and a perturbation range $\beta^\epsilon_\alpha(s)$, the attack is stealthy if $\rho(\bar{s}'_t, \phi_g) > \rho(\bar{s}_t, \phi_g)$.

The concept of stealthiness, as defined in previous studies, takes on various perspectives. For example, [36] characterizes it as the range of perturbations around the original observation. On the other hand, for those works that focus on the system safety [15, 17, 22], stealthiness is assessed in systems equipped with a detector, which implies avoiding detection. The work by [34] introduces an additional level of stealthiness called reward

stealthiness. They consider the reward stealthiness as 'the agent might easily detect a dramatic reward drop', which inspires us that, in the CPS domain, if there is a huge drop in the robustness of ϕ_g, the system may notice the anomaly behavior and detect that there is an adversary.

We add a new dimension of stealthiness within the context of STL-guided safe RL. The Definition 4.2 considers an attack as more stealthy if it can maintain the robustness value of ϕ_g from Equation 2 after the attack. Therefore, it cannot be detected by monitoring the robustness score of ϕ_g. Additionally, we introduce the perturbation set $\beta_p^\epsilon(s)$ to confine s' within specified bounds, thus delineating that the perturbation in observation adheres to established standards of stealthiness, as prior literature [20, 34]. In general, the problem is that the adversary wants to find the observation perturbation s'_t to force the system to take a malicious action u' which minimizes the robustness of the safety specification $\rho(\bar{s}_{t+1}, \phi_c)$ bounded by the stealthiness.

$$\begin{aligned} s'_t &= \underset{s'_t}{\operatorname{argmin}}\, \rho(\bar{s}_{t+1}, \phi_c) \\ \text{s.t. } \| s'_t - s_t \|_\alpha &\leq \epsilon \\ \rho(\bar{s}'_t, \phi_g) &> \rho(\bar{s}_t, \phi_g) \\ u'_t &= \pi(s'_t) \\ s_{t+1} &= p(s_t, u'_t) \end{aligned} \tag{7}$$

While adversary attacks directed at RL-based control have been extensively researched, our specific problem remains distinct and relatively unexplored. Previous studies have primarily concentrated on manipulating system observations to reduce the overall rewards, primarily impacting agent performance. These approaches often do not account for the crucial safety constraints of the system. We denote these methods as reward(value) decreasing (RD) methods and we show these methods can't achieve attack effectiveness.

Theorem 4.3. *Suppose there is a RD adversary policy h_{rd} method manipulates the observation as $s'_t = s_t + h_{rd}(s_t)$. The adversary policy h_{rd} cannot guarantee to achieve attack effectiveness.*

We provide proof for the Theorem 4.3 in subsection 4.3. To address the problem, we propose the **Safety Violation Attack (SVA)** framework where the adversary deliberately forces the system to violate the safety constraint under the limitation of stealthiness.

4.2 Safety Violation Attack Framework

White-box attack. We begin with the white-box attack. Since the adversary knows the transition function p and control policy π, the process can be formalized as an optimization problem as below:

However, directly solving the optimization function in Equation 7 is hard since an NN-based control policy is typically nonlinear and nonconvex [9]. We construct an alternate way of solving the s'_t. We divided Equation 7 into two parts. First, the attacker initiates the process by obtaining a malicious action u'_t which is designed to compromise the robustness of the safety constraint in the STL specification:

$$\begin{aligned} u'_t &= \underset{u'_t}{\operatorname{argmin}}\, \rho(\bar{s}_{t+1}, \phi_c) \\ \textit{where} \quad s_{t+1} &= p(s_t, u'_t) \end{aligned} \tag{8}$$

The u'_t serves as a targeted action to guide the subsequent observation perturbation. The next step involves executing the observation perturbation to induce the system to perform action u'_t. Then the observation perturbation s'_t can be generated using solvers like FGSM [16] and PGD [36] by minimizing $\ell(u_t, u'_t)$ where ℓ is a distance function that measures the distance between current action u_t with the adversary desired u'_t. We present our SVA framework under the white-box setting in Algorithm 1.

Algorithm 1 SVA (White-box version)

1: **Input:** the observation s_t, control policy π, STL specification ϕ_g and ϕ_c, distance function ℓ, update budget n, step size η, attack bound ϵ.
2: **Output:** observation perturbation s'_t
3: $u'_t \leftarrow \text{argmin}\ \rho(\bar{s}_{t+1}, \phi_c)$
4: $\Gamma(s_t) \leftarrow \{s'_t | \rho(\bar{s}'_t, \phi_g) > \rho(\bar{s}_t, \phi_g)\}$
5: $B(s_t) \leftarrow \beta^{\epsilon}_{\alpha}(s_t) \cap \Gamma(s_t)$
6: **for** $i = 0 : n$ **do**
7: $\quad u_t = \pi(s'_t)$
8: $\quad grad = \nabla_{s'_t} \ell\,(u_t, u'_t)$
9: $\quad s'_t = s'_t - \eta * \epsilon * sign(grad)$
10: $\quad s'_t \leftarrow \text{Proj}_{B(s_t)}\,[s'_t]$
11: **end for**
12: **return** s'_t

Line 3 in Algorithm 1 calculates the optimal action u'_t that maliciously forces the system to violate ϕ_c. Line 4 computes the $\Gamma(s_t)$, which is the set of s' that is constrained by stealthiness. Line 5 gets the final admissible set $B(s_t)$, which is the intersection of $\Gamma(s_t)$ and the set of the perturbation range $\beta^{\epsilon}_{\alpha}(s_t)$. Note that the set $\Gamma(s_t)$ is available because we have assumed the adversary knows the predefined STL specification. Line 7 computes the current action u_t, and line 8 obtains the gradient of the 2-norm pairwise distance between u_t and u'_t to the s'_t. Line 9 iteratively updates the s'_t, and line 10 projects the s'_t within the admissible set $B(s_t)$. Finally, the algorithm returns the observation perturbation s'_t at time t.

Grey-box attack and black-box attack. We consider the situation when the adversary has no knowledge of one of these two or has no knowledge of both defined in Section 3.

The grey-box attacks refer to the scenario in Section 3 where the transition function or the control policy is unknown, the black-box attack scenarios assume both the transition function and the control policy are unknown.

In cases where the adversary lacks knowledge of the control policy, but has access to the transition function, a common solution is to train a surrogate control policy π' to substitute the π. This approach has been widely known [20, 41] as the transferability for an adversarial attack on supervised learning neural networks and RL policy. Since we assume the adversary can access the environment and the STL specification, the adversary can train such a surrogate control policy π' without knowing the origin control policy's algorithm and parameters.

In cases where the transition function p is not available to the adversary, it is infeasible to obtain a malicious action u' by solving equation 8. Existing studies like [48, 55] form this to a MDP problem and apply RL to train the adversary model to obtain an adversarial control input $u'_t = \pi_{adv}(s_t)$. The adversary's objective is to reduce the reward earned by the system. Therefore, an adversary policy is trained using the reward function $\hat{r}_t = -r_t$ [55] where r_t is the reward function of the victim policy. Instead of reducing the reward to degrade the control performance, we propose an alternative approach that leverages the control policy and reverses safety constraints.

Definition 4.4 (Safety Violated Adversary model). Given a CMDP $M := (S, A, p, r, c)$ and an STL-guided RL policy π with STL specification $\phi = F_{[t_0, t_0+T]}\phi_g \wedge G_{[t_0, t_0+T]}\phi_c$, a safety violated adversary policy π_{adv} can be trained using the reversed version of the safety specification $F_{[t_0, t_0+T]}\phi_{adv}$ where $\phi_{adv} = \neg\phi_c$. The adversary policy can always obtain the malicious action $u'_t = \pi_{adv}(s_t)$ that forces the system to violate the safety constraint.

The Definition 4.4 gives a solution to obtain the malicious action when the adversary does not access the transition function. The malicious action u_t' is used to compute the s_t' as the Algorithm 1 lines 4-8 does. We provide the details of the algorithm in Algorithm 2.

When the attacker lacks both the knowledge of the transition function and control policy, we refer to it as a black-box scenario. In this case, the adversary can employ both the aforementioned methods (surrogate control policy and adversary model) to implement the SVA framework.

Algorithm 2 SVA (Black-box version)

Input: the current system state s_t, STL specification ϕ_g and ϕ_c, surrogate control policy π', adversary model π_{adv}, distance function ℓ, update budget n, step size η
Output: observation perturbation s_t'
$u_t' \leftarrow \pi_{adv}(s_t)$
$\Gamma(s_t) \leftarrow \{s_t' | \rho(\bar{s}_t', \phi_g) > \rho(\bar{s}_t, \phi_g)\}$
$B(s_t) \leftarrow \beta_\alpha^\epsilon(s_t) \cap \Gamma(s_t)$
for $i = 0 : n$ **do**
    $u_t = \pi'(s_t')$
    $grad = \nabla_{s_t'} \ell(u_t, u_t')$
    $s_t' = s_t' - \eta * \epsilon * sign(grad)$
    $s_t' \leftarrow \text{Proj}_{B(s_t)}[s_t']$
end for
return s_t'

4.3 Theoretical Analysis

We conduct a comparative analysis between the proposed Safety Violation Attack (SVA) and the reward decreasing (RD) attack introduced by existing works [55, 56]. We demonstrate an actor-critic RL algorithm to show that reward-decreasing (RD) attacks can't violate the safety specification effectively. The underlying intuition of the RD attack is to induce a sub-optimal action characterized by a lower observation-action value function $Q(s, a)$ output by the critic network. Next, we explain why SVA achieves better attack performance than RD attacks.

We denote h_{rd} as the reward-decrease adversary algorithm. The adversary h_{rd} is a group of methods, such as researchers in [56] leverage the gradients of the critic network to guide the observation adversary towards minimizing the value function $Q(s, a)$. We start with the value function and a basic victim policy π where the policy is trained with the STL specification from Equation 4. Despite different RD attack algorithms, the value function under such an adversary can be obtained as follows:

Definition 4.5. (Value function under adversarial dynamics). Given a victim control policy $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ and adversary algorithm $h_{rd} \in h : \mathcal{S} \to \mathcal{S}'$, we get

$$\tilde{V}_{\pi \circ h}(s) = \mathbb{E}_{\pi \circ h} \sum_{t=0}^{T} \gamma^t \rho(\bar{s}_t, \Phi)$$

Theorem 4.6 (SVA Is Effective for STL-Guided Safe RL). *The Safety Violation Attack (SVA) is more effective in violating safety than reward-decreasing (RD) attacks when targeting STL-guided safe RL policies.*

Proof. Let $\tilde{V}_{\pi \circ h}(s)$ be the value function under an adversary h that perturbs the state s to an observed state s':

$$\tilde{V}_{\pi \circ h}(s) = \mathbb{E}_{\pi \circ h}\left[\sum_{t=0}^{T} \gamma^t \rho(\bar{s}_t, \Phi)\right], \tag{9}$$

If we assume $\Phi = F_{[0,t]}\phi_g \wedge G_{[0,t]}\phi_c$, then the robustness at time t is:

$$\rho(\bar{s}_t, \Phi) = \min\left(\underbrace{\max_{t' \in [0,t]} \rho(\bar{s}_{t'}, \phi_g)}_{A_t}, \underbrace{\min_{t' \in [0,t]} \rho(\bar{s}_{t'}, \phi_c)}_{B_t}\right). \tag{10}$$

Suppose the agent initially satisfies the safety constraint ϕ_c but not the goal ϕ_g, i.e., $B_t > 0$ and $A_t < 0$. In this case, without any attack, the adversarial objective simplifies to:

$$\tilde{V}_{\pi}(s) = \sum_{t=0}^{T} \gamma^t A_t. \tag{11}$$

Now consider a reward-decreasing (RD) attacker that perturbs the current observation s_t to s_t' such that:

$$h_{rd}(s_t) = \arg\min_{s_t'} \rho(\bar{s}_t', \Phi) = \arg\min_{s_t'} \min(A_t, B_t). \tag{12}$$

Given $A_t < 0$ and $B_t > 0$, we get:

$$\min(A_t, B_t) = A_t \quad \Rightarrow \quad h_{rd}(s_t) = \arg\min_{s_t'} A_t. \tag{13}$$

That is, the RD attacker aims to decrease A_t, which relates to goal completion, not safety violation. Hence, the RD attacker may hinder goal achievement but does not directly induce safety violations.

In contrast, our SVA attacker specifically targets the safety specification ϕ_c:

$$h_{sva}(s_t) = \arg\min_{s_t'} \rho(\bar{s}_t', \phi_c) = \arg\min_{s_t'} B_t. \tag{14}$$

Therefore, SVA focuses on minimizing the robustness of the safety constraint. While we cannot guarantee SVA will cause a violation for every trajectory, empirical results in Section 5 show that SVA is more effective at violating safety in STL-guided safe RL than RD methods.

Comparison with Optimization-based Safe RL. In contrast, for optimization-based safe RL (e.g., CPO), the objective function incorporates both reward and safety cost:

$$\max_{\pi} \mathbb{E}^{\pi}\left[\sum_{t=0}^{T-1} \gamma^t r(s_t, u_t, s_{t+1}) - \lambda \sum_{t=0}^{T-1} \gamma^t c(s_t, u_t, s_{t+1})\right], \tag{15}$$

where λ is a penalty multiplier and $c(\cdot)$ is the cost function. An RD attacker in this setting perturbs observations to reduce the total reward and/or increase the cost:

$$h_{rd}(s_t) = \arg\min_{s_t'} \mathbb{E}_{s_{t+1} \sim p(\cdot|s_t', u_t)}[r(s_t', u_t, s_{t+1}) - \lambda c(s_t', u_t, s_{t+1})]. \tag{16}$$

This directly impacts both the task and safety objectives, making RD attacks more effective in optimization-based safe RL than in STL-guided settings.

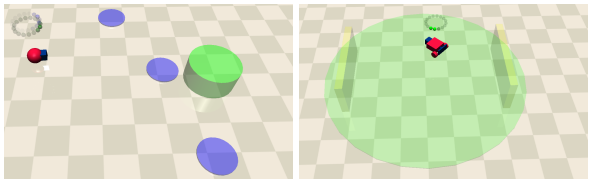

Fig. 1. The PointGoal (left) and CarCircle (right) benchmarks.

Thus, for STL-guided safe RL, the RD attacker tends to decrease task completion without necessarily violating safety. SVA, by contrast, targets the safety constraint, making it more suitable for violating safety under STL-based supervision. □

Moreover, although the SVA is designed over specification violation, we show that the SVA can also be effectively applied to optimization-based safe RL algorithms in Section 5.4, due to the behavior alignment with STL-guided safe RL.

5 Experiments

This section outlines our experimental methodology for evaluating the SVA framework across various benchmarks and safe RL algorithms.

5.1 Benchmarks

Safety Gym. We first perform experiments on the OpenAI platform, specifically using the Safety Gym environment [42] PointGoal and CarCircle. The PointGoal task is to control the point to reach the goal (green) while avoiding the hazard (purple) shown in Fig. 1. The CarCircle platform is to control the car navigation inside the green circle and avoid colliding with the wall (yellow).

For the PointGoal benchmark, the Point has sensors to observe the distance to the goal and the unsafe region. We set one goal and three hazards in the environment. We denote the d_g and d_c as the distance to the goal and the closest unsafe region. We define the STL specification for the task as below:

$$\Phi = F(d_g < r_g) \wedge G(d_c > r_c)$$

Where the r_g and r_c are the radius of the goal and hazards. Note that the PointGoal is a typical reach-avoid task. We use the dense reward function from existing work [18] that returns the robustness value every time step. The reward function is defined as:

$$R_t = \min_{t' \in [0,t]} \left(\max_{t' \in [0,t]} (r_g - d_g), \min_{t' \in [0,t]} (d_h - r_h) \right)$$

The original reward function in the CarCircle benchmark incentivizes the car to accelerate while ensuring it stays within the circular track. However, the benchmark lacks a specific directive regarding the car's velocity, as STL cannot explicitly formalize this requirement. To address this, we introduce an additional criterion: the car must attain a predefined baseline velocity and maintain this speed once achieved. We consider that the car colliding with the wall is the safety constraint. We simulate this task in accordance with the STL specification as outlined below:

$$\Phi = F(\frac{v}{|r_{car} - r_{circle}|} > v_0) \wedge G(d_c > 0)$$

Where v is the current velocity of the car and v_0 is the required velocity that the car would reach. r_{car} is the distance of the car to the center of the circle. r_{circle} is the radius of the circle. $d_c > 0$ is the requirement that the

ϵ	PointGoal							CarCircle						
	WB	GB-C	GB-P	BB	GA	LAA	MAD	WB	GB-C	GB-P	BB	GA	LAA	MAD
0.01	5.6%	4.6%	5.4%	5.2%	2%	3.6%	3.6%	23.2%	14%	14.8%	12.8%	0%	5.6%	6.8%
0.05	14%	12%	10%	9.8%	6%	4.8%	4.8%	48.8%	16.4%	17.2%	15.6%	3.4%	7.6%	8.4%
0.10	45%	34.2%	30.6%	20.8%	7.2%	6%	7.2%	84.2%	24.8%	21%	16.8%	9.2%	8%	9.2%
0.15	74.6%	62.4%	52%	38.6%	7.6%	6.8%	11.4%	90.4%	32.6%	26.8%	19.2%	16.8%	12%	10.4%
	DC Motor							Bicycle						
0.01	18.8%	13.2%	15.6%	13.2%	5.6%	8.8%	9.6%	18.4%	16.4%	17.4%	13%	7.2%	8.8%	10.2%
0.05	19.4%	14.8%	16.2%	14.8%	6.4%	12.8%	14%	32.2%	24.6%	23.6%	20%	12.6%	12.6%	13.6%
0.10	33.6%	28.4%	22.8%	19%	13.2%	16.8%	16%	44.8%	38%	35%	25.8%	14%	16.8%	18.2%
0.15	46%	41.2%	24.4%	22.4%	18%	21.2%	16.4%	46.2%	38.6%	35.6%	28.8%	23%	21.2%	20%

Table 1. Attack performance measured by the violation rate. ϵ is the perturbation range. The WB, GB-C, GB-P, and BB represent our SVA framework under the white box, a grey box with a known control policy, a grey box with a known transition function, and a black box, respectively. GA, LAA, and MAD are the three baseline methods introduced in the former subsection. The higher the percentage, the better the attack performance to violate the safety.

car should keep a distance from the wall. We generate the reward function as:

$$R_t = \min_{t' \in [0,t]} \left(\max_{t' \in [0,t]} \left(\frac{v}{|r_{car} - r_{circle}|} - v_0 \right), \min_{t' \in [0,t]} (d_c - r_c) \right)$$

For the two benchmarks, we train the control policy π using PPO [43] for 10 million steps, and the reach rate for the control point is greater than 95% with less than 2% violation rate.

Classical control system. We also perform the SVA framework on two classical control systems from the CPS community: DC Motor Position [57], Bicycle [25]. DC Motor position controls the motor angle to a desired position by using the current as control input. The Bicycle has two control inputs: acceleration and steering angle, with the goal of speed and steering angle.

We establish a control objective along with two designated unsafe regions for each benchmark. For instance, in the case of the DC Motor position, we define a target motor angle as the objective and designate two other motor angles as unsafe thresholds. The control policy's task is to navigate the angle toward the objective while steering clear of unsafe angles within a predefined time horizon. We quantify the goal and unsafe regions using the 2-norm Euclidean distance metric.

To train the control policy, we employ the Soft Actor-Critic (SAC) algorithm. The training process is carried out for 500,000 iterations, utilizing the reward function, which is the robustness of the STL specification similar to Equation 17. Both of the control policies have a greater than 96% reach rate and have less than 1% violation rate.

5.2 Experiment Setting

We first introduce the SVA performance details under different adversary knowledge levels for the SVA framework. Then we demonstrate the baseline methods for comparison.

SVA framework setting. We set up SVA methods with different levels of adversary knowledge: White box(WB), grey box with known control policy (GB-C), grey box with known transition function (GB-P), and black box (BB). For all the experience, the set of perturbation set $\beta_\alpha^\epsilon(s)$ is defined as an l_∞ norm ball around s with the budget ϵ. We set different time horizons T for the four benchmarks: 10,000 steps for PointGoal, 5,000 for CarCircle, and 50 for DC Motor and Bicycle. The adversary is considered to be effective when it can violate the

ϵ	PointGoal							CarCircle						
	WB	GB-C	GB-P	BB	GA	LAA	MAD	WB	GB-C	GB-P	BB	GA	LAA	MAD
0.01	86.4%	89.6%	89.6%	89.6%	93.2%	91.2%	93.6%	72%	84.4%	84%	86.4%	98%	92%	91.6%
0.05	82.8%	86.4%	86.4%	87.6%	91.2%	90%	90.8%	51%	81.6%	79.8%	82.6%	93.4%	89.6%	88%
0.10	52%	60%	80.8%	69.6%	90%	88%	90%	11.8%	71.6%	75.6%	80.4%	86.6%	87.4%	84.6%
0.15	19.6%	32.2%	42.4%	50.4%	88%	84%	85.4%	4.8%	62.6%	70.8%	77.4%	81%	82.8%	83.2%
	DC Motor							Bicycle						
0.01	76%	79.2%	79.2%	79.6%	82%	86.8%	85.2%	73.6%	79.8%	78.2%	83%	90.6%	86.8%	85.8
0.05	75.2%	80%	80%	80.4%	81.8%	82.4%	82.4%	64%	71.8%	73.2%	77%	85.2%	82.4%	78.6
0.10	56.8%	61.2%	73.6%	76%	72.4%	77.2%	82%	48.6%	59.4%	62%	72.4%	80.8%	77.2%	77.4
0.15	45.8%	48.4%	64.4%	68.4%	64.4%	66.8%	76%	46.2%	57%	35.6%	60.2%	72%	66.8%	76.8

Table 2. The reach rate for each benchmark with different adversary algorithms. The lower percentage represents that the adversary has more impact on the task completion.

safety constraint within the predefined time horizon T. We use the SAC algorithm to train the surrogate control policy and PPO to train the adversary model for grey-box and black-box attacks.

Gradient-based attack. Gradient-based Attack (GA) refers the group of methods from [38, 56]. GA minimizes the value function of the critic $Q^{\text{target}}(s, a)$. It first gets the gradient: $grad = \nabla_s Q^{\text{target}}(s, a)$ and updates the potential adversarial state as $s_i = s - n_i * \frac{grad}{\|grad\|}$ where n_i is the sampled noise.

Learning-based adversary attack. The learning-based adversary attack (LAA) [56] leverages the idea that a control policy π with an observation adversary can be formalized to an SA-MDP. The adversary's task can also be viewed as solving an MDP $\hat{M} = (\mathcal{S}, \hat{A}, \hat{p}, \hat{r}, \gamma)$ where $\hat{r}(\cdot \mid \cdot) = -r(\cdot \mid \cdot)$. The intuition behind this is that the adversary policy wants to reduce the reward obtained by the victim system, resulting in a negative reward for the adversary when the victim system obtains a positive reward.

Maximal action difference attack. Some works consider decreasing the RL return reward by attacking the observation, resulting in the system taking suboptimal action. We consider the Maximal Action Difference Attack (MAD) from [56], which has proved to be efficient and simple. The MAD obtains the s'_t by minimizing: $L_{\text{MAD}}(s') := -D_{\text{KL}}(\pi(s) \| \pi(s'_t))$ where D_{KL} is the KL-divergence.

Note that for a fair comparison, all three baseline observation perturbations are constrained by the stealthiness requirement. Additionally, the three baseline methods are under a black box setting. For example, the GA uses the surrogate control policy to compute the gradient to keep in the black box scenario. This ensures that each method is evaluated under consistent conditions, allowing for meaningful comparisons of their effectiveness.

5.3 Results

We first present the attack results of our SVA and three baseline methods across each benchmark. For each benchmark, we evaluate SVA using 500 experiments with random initial points, and we report the percentage of safety violations within the time horizon T as a function of the perturbation range ϵ, as shown in Table 1. Each episode ends when the system either reaches the goal or violates the safety constraints. The perturbation range ϵ is selected from the set $\epsilon \in [0.01, 0.05, 0.10, 0.15]$. The results demonstrate that even with a small perturbation range, the observation perturbations are still effective.

Observation 1: The white-box SVA demonstrates the highest rate of violations, outperforming all the other methods. The SVA can successfully force the victim system into unsafe conditions even when the perturbation range $\epsilon = 0.01$. Notably, the WB SVA can achieve a 90.4% violation rate with $\epsilon = 0.15$ for CarCircle, while all three baseline methods only have less than 17% violation rate. The black-box SVA has the worst performance within

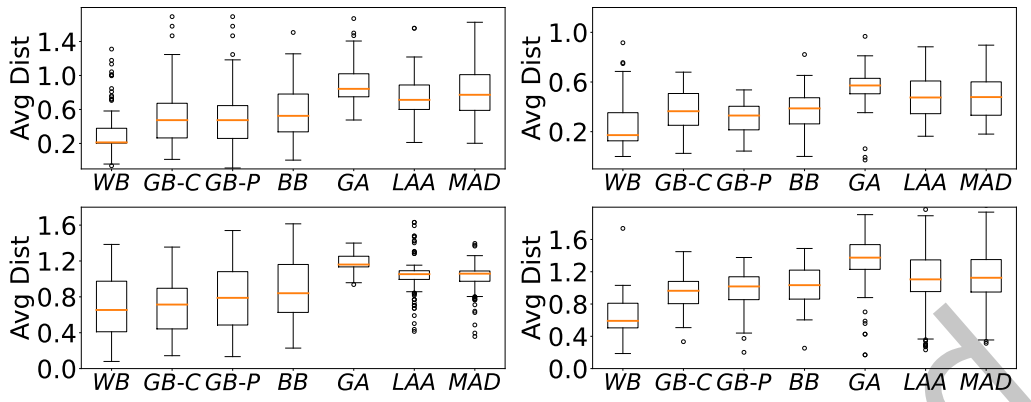

Fig. 2. The average distance to unsafe for the SVA with four benchmarks under the white box (WB), a grey box with control policy (GB-C), a grey box with transition function (GB-P), and a black box (BB), and three baseline attacks. Top left: PointGoal, Top right: CarCircle, Bottom left: DCMotor, Bottom right: Bicycle.

the four SVA scenarios but still outperforms the three baselines, which meets our expectations. The performance difference between BB and WB shows that obtaining knowledge of the transition function and control policy indeed increases the possibility of violating safety. This observation also proves that the transition function and control policy are useful to the adversary.

Note that although the SVA does not specifically target a decrease in the reach rate, the SVA still has a significant impact on the reach rate. The WB achieves the highest performance in reducing the reach rate. The WB attack precisely found the vulnerability of the system, making the system incapable of reaching its goal and violating safety.

Observation 2: There is an intriguing observation that when ϵ is small, GB-P demonstrates better efficiency than GB-C. However, this trend does not hold for larger ϵ values, where GB-C is shown to be more efficient. We illustrate the observation using the following claim:

- The adversary knows the control policy means that the generated observation perturbation s_t' is more potent in reducing the robustness of the safety constraint compared to the action obtained through the adversary model π_{adv} used in the GB-P and BB setting.
- When the adversary knows the control policy, it signifies that the produced observation perturbation s_t' can induce the system to take an action that is closer to the adversary's intended action, while a surrogate control policy may lead to an action that is not as closely aligned, potentially due to neural network transferability problem. The adversarial perturbation can not always be directly transferred to another policy with different algorithms and parameters.

When the perturbation range is small, it becomes difficult for the adversary to generate a corresponding observation perturbation s_t' for a given malicious action u_t'. This reduces the importance of knowing the control policy, making GB-C less effective compared to GB-P. However, when the perturbation range is large, knowledge of the control policy becomes critical. It allows the generation of s' that leads to the precise malicious action, thereby enhancing the attack's effectiveness. Although GB-P produces more severe actions, the observation perturbations generated by the surrogate control policy still lack essential information.

Observation 3: Fig. 2 presents the average distance to unsafe states across 100 experiments using four SVA methods with $\epsilon = 0.1$. The figure indicates that the WB (white-box) scenario results in the shortest average distance, meaning that WB has the highest probability of driving the system toward unsafe states. Conversely,

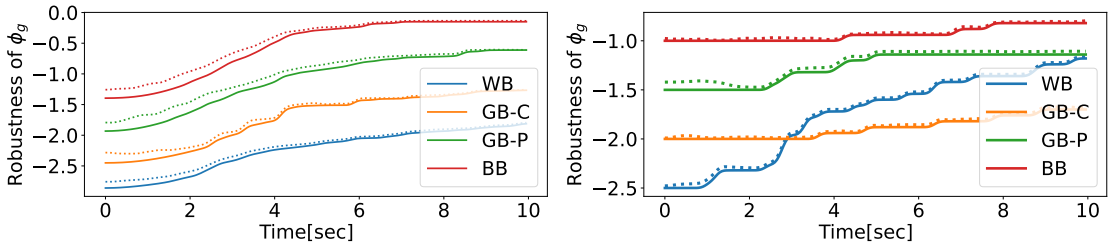

Fig. 3. The stealthiness measured by the observed robustness value of the goal ϕ_g. The dotted line is the observed robustness after the attack, the solid line is the robustness before the attack. The dotted lines in the four attack scenarios are all greater than the solid line, which means that our SVA maintains stealthiness.

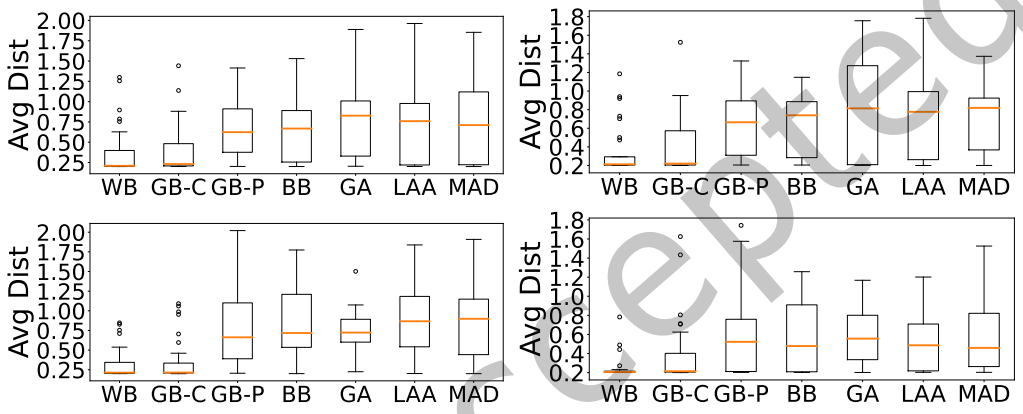

Fig. 4. The average distance to unsafe regions for the SVA with the PointGoal task, using four optimization-based algorithms under different attack methods. Top left: PPOLag, Top right: CPO, Bottom left: CVPO, Bottom right: FOCOPS.

BB (black box) yields the greatest average distance, suggesting weak attack performance across the four SVA scenarios, consistent with the result observed in Observation 1.

Observation 4: We verified the stealthiness of our SVA and present the results in Fig. 3, using the PointGoal and CarCircle benchmarks. Each figure illustrates four different observation histories across various attack scenarios. In each case, we assume the victim system uses the manipulated observations to calculate the robustness of the goal ϕ_g. The dotted line represents the robustness value after the attack, which remains consistently equal to or greater than the solid line. This confirms that all four SVA scenarios preserve their stealthiness.

5.4 Compatibility Analysis

We extend our SVA framework to attack other state-of-the-art safe RL algorithms to show that the SVA can also be effective for optimization-based safe learning algorithms. We additionally compare our method with the Maximum-cost (MC) and Maximum-reward (MR) attack from [34] with a black-box setting by training a surrogate Q network using PPOLag for generating attacks for MR and MC. We train four control policies using the PPOLag [45], CPO [2], FOCOPS [59], and CVPO [33] algorithms and present the attack results under the same settings as described in the previous subsection on PointGoal and CarCirlce tasks. The chosen safe RL algorithms are commonly used for comparison on Safety Gym benchmarks in several papers [33, 52, 53, 59].

		PointGoal				CarCircle			
Attacks	ϵ	**PPOLag**	**CPO**	**CVPO**	**FOCOPS**	**PPOLag**	**CPO**	**CVPO**	**FOCOPS**
BB	0.01	10.2%	20.8%	4.4%	11.8%	9.2%	6.4%	5.8%	7.0%
	0.05	11.6%	27.6%	10.4%	18.4%	14.6%	12.8%	11.4%	13.2%
	0.10	17.4%	28.8%	13.6%	24.8%	19.4%	15.6%	14.8%	16.2%
	0.15	19.8%	26.2%	19.2%	27.4%	22.0%	18.4%	17.6%	19.2%
WB	0.01	19.8%	25.0%	12.4%	18.6%	22.2%	15.2%	14.6%	16.8%
	0.05	35.4%	35.2%	29.2%	30.8%	29.4%	28.6%	27.8%	29.2%
	0.10	64.2%	51.6%	67.4%	60.8%	47.6%	43.2%	42.4%	45.0%
	0.15	72.6%	67.4%	80.4%	71.8%	65.8%	56.4%	55.2%	58.8%
GB-C	0.01	16.4%	25.4%	11.6%	15.6%	20.2%	14.8%	13.6%	15.4%
	0.05	25.2%	21.6%	27.8%	26.2%	19.8%	22.0%	21.4%	23.2%
	0.10	43.6%	53.4%	52.6%	55.2%	41.4%	38.6%	37.8%	40.2%
	0.15	76.2%	54.6%	75.4%	63.8%	49.0%	47.2%	46.6%	48.8%
GB-P	0.01	12.4%	23.6%	7.6%	13.8%	10.8%	8.4%	7.8%	9.2%
	0.05	13.6%	25.0%	13.4%	17.2%	14.6%	10.6%	9.8%	11.4%
	0.10	15.6%	30.6%	19.4%	28.8%	26.6%	20.4%	19.6%	21.2%
	0.15	22.4%	31.8%	22.6%	31.6%	33.8%	26.4%	25.8%	27.4%
GA	0.01	7.0%	9.6%	4.6%	8.2%	6.2%	5.6%	5.0%	6.0%
	0.05	9.6%	11.8%	7.4%	10.4%	7.4%	6.8%	6.2%	7.0%
	0.10	12.4%	14.2%	10.2%	13.6%	10.0%	8.4%	8.0%	9.2%
	0.15	15.0%	18.0%	12.6%	16.8%	12.8%	11.4%	10.8%	12.2%
LAA	0.01	8.0%	10.4%	5.6%	9.0%	6.4%	6.0%	5.4%	6.4%
	0.05	10.2%	12.6%	7.4%	11.2%	8.4%	7.8%	7.0%	8.2%
	0.10	13.2%	15.4%	12.2%	14.0%	10.8%	9.6%	9.2%	10.6%
	0.15	16.0%	19.0%	15.2%	17.4%	13.6%	11.8%	11.4%	12.8%
MAD	0.01	6.0%	8.0%	2.6%	7.4%	5.0%	4.6%	4.2%	5.4%
	0.05	7.6%	9.6%	6.4%	8.8%	6.2%	5.8%	5.4%	6.6%
	0.10	9.8%	12.0%	10.2%	11.4%	8.4%	7.6%	7.0%	8.2%
	0.15	12.4%	15.0%	13.2%	14.0%	10.4%	9.2%	8.6%	10.0%
MC	0.01	12.9%	21.8%	10.9%	13.3%	10.3%	9.9%	18.4%	19.5%
	0.05	17.3%	24.7%	24.8%	20.9%	23.7%	19.7%	23.7%	26.8%
	0.10	20.5%	35.3%	28.2%	38.8%	47.5%	43.0%	36.3%	39.6%
	0.15	23.6%	38.4%	33.1%	41.6%	54.3%	47.6%	47.0%	50.4%
MR	0.01	10.2%	18.6%	11.0%	11.9%	11.0%	9.1%	11.4%	13.0%
	0.05	15.2%	23.5%	20.6%	15.2%	15.9%	16.8%	20.1%	20.5%
	0.10	16.9%	26.5%	22.0%	23.7%	16.2%	18.3%	22.1%	25.4%
	0.15	20.9%	32.0%	29.3%	29.8%	22.2%	24.0%	24.2%	28.1%

Table 3. Violation probabilities for PPOLag, CPO, CVPO, and FOCOPS algorithms across two tasks (PointGoal and CarCircle) and attack types.

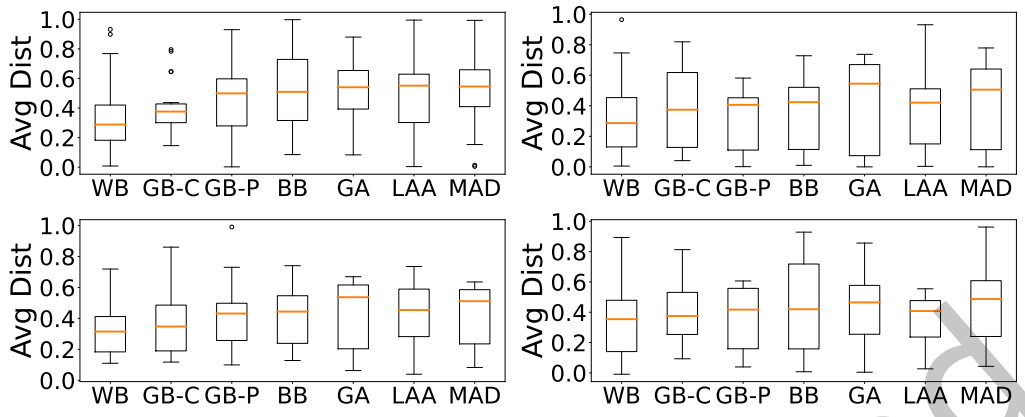

Fig. 5. The average distance to unsafe for the SVA with Carcircle task using four optimization-based algorithms under the different attack methods. Top left: PPOLag, Top right: CPO, Bottom left: CVPO, Bottom right: FOCOPS.

Observation 5: Table 3 reinforces the observation made during attacks on STL-guided safe RL. The white-box attack demonstrates the highest efficiency across both tasks. Similarly, GB-C performs better than GB-P, highlighting the importance of the adversary having knowledge of the control policy. Conversely, this also demonstrates that using a surrogate model for transferability degrades the effectiveness of the attack. The black-box SVA demonstrates the poorest attack performance. However, it is still better than the three baselines, which illustrate that our SVA framework can effectively perform safety violation attacks compared to other baseline attack methods, even without the system knowledge.

An interesting finding is that the four optimization-based safe RL methods exhibit similar vulnerabilities to those observed in STL-guided safe RL. The results in Table 3 show comparable attack performance under identical attack settings (attack type and ϵ). For instance, both the optimization-based and STL-guided safe RL algorithms achieve nearly a 70% violation rate when subjected to a white-box attack with $\epsilon = 0.15$, as seen in Tables 3 and 1. This reveals that both threads of safe RL algorithms are comparable and vulnerable to the safety violation aimed perturbation attack.

Observation 6: Additionally, it is worth noting that the three baseline methods are less effective at violating safety constraints compared to the SVA framework. Figure 4 illustrates the average distance to unsafe states under different attack types, where the white-box SVA demonstrates the best performance in minimizing safety violations. However, the baselines perform better against optimization-based methods than against STL-guided safe RL, as seen by comparing Figure 4 with the upper-left figure in Figure 2 for the PointGoal tasks. This supports our hypothesis in Section 4.3 that reward-decreasing attacks are more effective at causing safety violations in optimization-based safe RL algorithms. This also highlights the vulnerability of these algorithms, where reducing the reward can lead to more significant safety violations. In contrast, STL-guided safe RL proves to be more resilient to reward-decreasing attacks.

5.5 Cart-Pole Case Study

In this subsection, we test whether our SVA can attack a system that does not satisfy the STL task in Eq. 4. The target is the Cart-Pole environment from Gym [49]: a pole is attached to a cart that moves on a track. The pole starts upright, and the controller keeps it balanced by pushing the cart left or right.

For the first 5 s the cart must stay close to centre ($|x| \leq 0.4$ m) and the pole may not tilt more than 0.2 rad. In addition, within the first 2 s the pole must become almost vertical ($|\theta| \leq 0.05$ rad) and hold that pose for at least 1 s. Formally,

$$\Phi = G_{[0,5]}\big(|x| < 0.4 \wedge |\theta| < 0.2\big) \ \wedge \ F_{[0,2]}\, G_{[0,1]}\big(|\theta| < 0.05\big).$$

We train a PPO controller for 1×10^6 steps using the robustness of Φ as the reward. For the black-box SVA, we set the adversary's goal

$$\phi_{\text{adv}} = F_{[0,5]}\big(|x| > 0.4 \ \vee \ |\theta| > 0.2\big),$$

that is, the cart leaves the centre band or the pole falls.

Table 4 lists the satisfaction ratio under different attack methods and ϵ values. Because reward and safety are positively correlated for Cart-Pole, lowering the reward directly causes failure (pole falls or cart drifts). For this reason, LAA shows stronger attack performance than SVA, yet SVA still remains effective.

Table 4. Satisfaction ratio under different ε values for Cart-Pole system

Method	$\varepsilon = 0.01$	$\varepsilon = 0.02$	$\varepsilon = 0.03$	$\varepsilon = 0.04$
SVA(BB)	1.00	1.00	0.54	0.52
LAA	1.00	1.00	0.00	0.00
MAD	1.00	1.00	0.66	0.61
GA	1.00	1.00	0.78	0.57

6 Discussion

This paper introduces the SVA framework, specifically targeting STL-guided safe RL, and can also be effective against optimization-based safe RL. We argue that, for any safe RL algorithms, the SVA framework has the potential to drive the system into unsafe regions if the adversary has knowledge of both the safety constraints and the control policy. However, if the adversary lacks access to the control policy, training a surrogate policy becomes impractical due to the unknown reward function.

Limitations of the SVA framework. In the white-box setting of the SVA framework, we assume the adversary uses the transition function p to determine the malicious action at each time step. However, this approach is suboptimal because it only achieves the optimal action for the specific step without considering long-term effects. A potential improvement would involve leveraging the transition function to compute a sequence of malicious actions, $u'_0, u'_1, ..., u'_t$. This sequence-based approach could be framed as a reachability problem, which has been explored in previous works [57]. However, these studies typically compute attack sequences based on a precise system model and model-based controller with bounded noise, which is beyond the scope of this paper.

Another limitation of our work is the restricted form of the STL specifications we consider, which are of the form $\Phi = F_{[t_0,t_0+T]}\phi_g \wedge G_{[t_0,t_0+T]}\phi_c$. While this form captures many practical tasks with a single goal and safety constraint, it does not generalize to more complex scenarios. For example, control tasks that require achieving multiple goals or maintaining goal satisfaction throughout the time horizon often lead to STL formulas that cannot be easily simplified into this conjunctive form. Our SVA method focuses on reducing the robustness of the overall STL specification. However, in cases where reducing the overall STL robustness does not significantly impact the robustness of the safety constraint ϕ_c, the effectiveness of the attack may be compromised.

Defense strategies. While the SVA framework demonstrates how malicious sensor attacks can lead a CPS to take hazardous actions, there are several ways to mitigate or defend against such attacks. Below, we discuss potential defenses against the Safety Violation Attack in CPSs.

1. Robust training: Robust training has proven effective in mitigating adversarial perturbations and enhancing policy resilience. Many robust training frameworks rely on adversarial techniques to generate attacks, allowing policies to be trained against these adversaries. We propose that training a policy robustly against the SVA adversary, specifically designed to target sensor attacks that threaten safety, could significantly improve both controller robustness and system safety.

2. Formal verification: In addition to focusing on safety during the training phase, another potential defense is identifying unsafe states using a prior model. Formal verification can establish the correctness of system behavior and identify unsafe states and actions, enabling the system to halt before an adversary drives it into a dangerous region. However, the success of this method depends on having an accurate system model. If the environment changes and the identification of unsafe states is incomplete or outdated, the system may remain vulnerable to threats.

3. Gradient information protection: Our attack framework generates perturbations by leveraging the gradient information of the control policy. Notably, different safe RL algorithms exhibit varying degrees of sensitivity to such gradient-based attacks. Optimization-based safe RL algorithms are particularly vulnerable to reward-decreasing attacks, where the exploitation of gradient information can significantly impact performance. In contrast, STL-guided safe RL tends to be more resilient in comparison. To mitigate the harmful effects of gradient leakage, it is crucial to conceal safety-related information within the gradient. This can help prevent adversaries from exploiting the gradient to degrade system performance or compromise safety.

7 Conclusion

In this study, we introduced the Safety Violation Attack framework, a novel approach for assessing the vulnerability of systems controlled by STL-guided safe RL algorithms. We implemented various attack strategies based on different levels of adversarial knowledge, demonstrating that existing adversarial attacks on RL are insufficient in effectively violating safety constraints. The SVA framework was evaluated across multiple benchmarks, including the Safety Gym platform. Our findings underscore the risks associated with deploying STL-guided RL and optimization-based safe RL controllers, particularly in safety-critical applications. The results reveal that the SVA framework is capable of identifying system vulnerabilities, emphasizing the need for stronger security measures in such environments.

Acknowledgments

This work was supported in part by NSF CNS 2442914 and CNS-2333980. 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] Houssam Abbas, Georgios Fainekos, Sriram Sankaranarayanan, Franjo Ivančić, and Aarti Gupta. 2013. Probabilistic temporal logic falsification of cyber-physical systems. *ACM Transactions on Embedded Computing Systems (TECS)* 12, 2s (2013), 1–30.

[2] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. 2017. Constrained policy optimization. In *International conference on machine learning*. PMLR, 22–31.

[3] Arend Aerts, Bryan Tong Minh, Mohammad Reza Mousavi, and Michel A Reniers. 2018. Temporal logic falsification of cyber-physical systems: An input-signal-space optimization approach. In *2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 214–223.

[4] Takumi Akazaki, Shuang Liu, Yoriyuki Yamagata, Yihai Duan, and Jianye Hao. 2018. Falsification of cyber-physical systems using deep reinforcement learning. In *Formal Methods: 22nd International Symposium, FM 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 15-17, 2018, Proceedings 22*. Springer, 456–465.

[5] Lukas Brunke, Melissa Greeff, Adam W Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P Schoellig. 2022. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*

(2022).
[6] Alberto Camacho, Rodrigo Toro Icarte, Toryn Q Klassen, Richard Anthony Valenzano, and Sheila A McIlraith. 2019. LTL and Beyond: Formal Languages for Reward Function Specification in Reinforcement Learning.. In *IJCAI*, Vol. 19. 6065–6073.
[7] Yulong Cao, Chaowei Xiao, Benjamin Cyr, Yimeng Zhou, Won Park, Sara Rampazzi, Qi Alfred Chen, Kevin Fu, and Z Morley Mao. 2019. Adversarial sensor attack on lidar-based perception in autonomous driving. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*.
[8] Hongkai Chen, Scott A Smolka, Nicola Paoletti, and Shan Lin. 2023. An STL-based approach to resilient control for cyber-physical systems. In *Proceedings of the 26th ACM International Conference on Hybrid Systems: Computation and Control*. 1–12.
[9] Yize Chen, Yuanyuan Shi, and Baosen Zhang. 2018. Optimal control via neural networks: A convex approach. *arXiv preprint arXiv:1805.11835* (2018).
[10] Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. 2018. A lyapunov-based approach to safe reinforcement learning. *Advances in neural information processing systems* 31 (2018).
[11] Yinlam Chow, Ofir Nachum, Aleksandra Faust, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. 2019. Lyapunov-based safe policy optimization for continuous control. *arXiv preprint arXiv:1901.10031* (2019).
[12] Alexandre Donzé and Oded Maler. 2010. Robust satisfaction of temporal logic over real-valued signals. In *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer.
[13] Scott Fujimoto, Edoardo Conti, Mohammad Ghavamzadeh, and Joelle Pineau. 2019. Benchmarking batch deep reinforcement learning algorithms. *arXiv preprint arXiv:1910.01708* (2019).
[14] Nathan Fulton and André Platzer. 2018. Safe reinforcement learning via formal methods: Toward safe control through proof and learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.
[15] Amin Ghafouri, Yevgeniy Vorobeychik, and Xenofon Koutsoukos. 2018. Adversarial regression for detecting attacks in cyber-physical systems. *arXiv preprint arXiv:1804.11022* (2018).
[16] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).
[17] Paul Griffioen, Sean Weerakkody, Bruno Sinopoli, Omur Ozel, and Yilin Mo. 2019. A tutorial on detecting security attacks on cyber-physical systems. In *2019 18th European control conference (ECC)*. IEEE, 979–984.
[18] Nathaniel Hamilton, Preston K Robinette, and Taylor T Johnson. 2022. Training agents to satisfy timed and untimed signal temporal logic specifications with reinforcement learning. In *International Conference on Software Engineering and Formal Methods*. Springer, 190–206.
[19] Mengdi Huai, Jianhui Sun, Renqin Cai, Liuyi Yao, and Aidong Zhang. 2020. Malicious attacks against deep reinforcement learning interpretations. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.
[20] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. 2017. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284* (2017).
[21] Jiaming Ji, Borong Zhang, Jiayi Zhou, Xuehai Pan, Weidong Huang, Ruiyang Sun, Yiran Geng, Yifan Zhong, Josef Dai, and Yaodong Yang. 2023. Safety gymnasium: A unified safe reinforcement learning benchmark. *Advances in Neural Information Processing Systems* 36 (2023).
[22] Amir Khazraei, Spencer Hallyburton, Qitong Gao, Yu Wang, and Miroslav Pajic. 2022. Learning-based vulnerability analysis of cyber-physical systems. In *ACM/IEEE 13th International Conference on Cyber-Physical Systems*. IEEE.
[23] Panagiota Kiourti, Kacper Wardega, Susmit Jha, and Wenchao Li. 2020. Trojdrl: evaluation of backdoor attacks on deep reinforcement learning. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.
[24] Fanxin Kong, Meng Xu, James Weimer, Oleg Sokolsky, and Insup Lee. 2018. Cyber-physical system checkpointing and recovery. In *ACM/IEEE 9th International Conference on Cyber-Physical Systems*. IEEE.
[25] Jason Kong, Mark Pfeiffer, Georg Schildbach, and Francesco Borrelli. 2015. Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE intelligent vehicles symposium (IV)*. IEEE, 1094–1099.
[26] Xiao Li, Yao Ma, and Calin Belta. 2018. A policy search method for temporal logic specified reinforcement learning tasks. In *Annual American Control Conference*. IEEE.
[27] Xiao Li, Cristian-Ioan Vasile, and Calin Belta. 2017. Reinforcement learning with temporal logic rewards. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE.
[28] Zeyang Li, Chuxiong Hu, Yunan Wang, Yujie Yang, and Shengbo Eben Li. 2024. Safe reinforcement learning with dual robustness. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024).
[29] Qingkai Liang, Fanyu Que, and Eytan Modiano. 2018. Accelerated primal-dual policy optimization for safe reinforcement learning. *arXiv preprint arXiv:1802.06480* (2018).
[30] Mengyu Liu, Pengyuan Lu, Xin Chen, Fanxin Kong, Oleg Sokolsky, and Insup Lee. 2023. Fulfilling Formal Specifications ASAP by Model-free Reinforcement Learning. *arXiv preprint arXiv:2304.12508* (2023).

[31] Mengyu Liu, Lin Zhang, Pengyuan Lu, Kaustubh Sridhar, Fanxin Kong, Oleg Sokolsky, and Insup Lee. 2022. Fail-safe: Securing cyber-physical systems against hidden sensor attacks. In *IEEE Real-Time Systems Symposium*. IEEE.
[32] Mengyu Liu, Lin Zhang, Vir V Phoha, and Fanxin Kong. 2023. Learn-to-respond: Sequence-predictive recovery from sensor attacks in cyber-physical systems. In *IEEE Real-Time Systems Symposium*. IEEE.
[33] Zuxin Liu, Zhepeng Cen, Vladislav Isenbaev, Wei Liu, Steven Wu, Bo Li, and Ding Zhao. 2022. Constrained variational policy optimization for safe reinforcement learning. In *International Conference on Machine Learning*. PMLR, 13644–13668.
[34] Zuxin Liu, Zijian Guo, Zhepeng Cen, Huan Zhang, Jie Tan, Bo Li, and Ding Zhao. 2022. On the robustness of safe reinforcement learning under observational perturbations. *arXiv preprint arXiv:2205.14691* (2022).
[35] Zuxin Liu, Zijian Guo, Zhepeng Cen, Huan Zhang, Yihang Yao, Hanjiang Hu, and Ding Zhao. 2023. Towards robust and safe reinforcement learning with benign off-policy data. In *International Conference on Machine Learning*. PMLR, 21586–21610.
[36] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* (2017).
[37] Oded Maler and Dejan Nickovic. 2004. Monitoring temporal properties of continuous signals. In *International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*. Springer.
[38] Anay Pattanaik, Zhenyi Tang, Shuijing Liu, Gautham Bommannan, and Girish Chowdhary. 2017. Robust deep reinforcement learning with adversarial attacks. *arXiv preprint arXiv:1712.03632* (2017).
[39] Diego GS Pivoto, Luiz FF de Almeida, Rodrigo da Rosa Righi, Joel JPC Rodrigues, Alexandre Baratella Lugli, and Antonio M Alberti. 2021. Cyber-physical systems architectures for industrial internet of things applications in Industry 4.0: A literature review. *Journal of manufacturing systems* (2021).
[40] Aniruddh G Puranic, Jyotirmoy V Deshmukh, and Stefanos Nikolaidis. 2021. Learning from demonstrations using signal temporal logic in stochastic and continuous domains. *IEEE Robotics and Automation Letters* 6, 4 (2021), 6250–6257.
[41] Yunxiao Qin, Yuanhao Xiong, Jinfeng Yi, and Cho-Jui Hsieh. 2023. Training meta-surrogate model for transferable adversarial attack. In *Proceedings of the AAAI Conference on Artificial Intelligence*.
[42] Alex Ray, Joshua Achiam, and Dario Amodei. 2019. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708* 7, 1 (2019), 2.
[43] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).
[44] Nikhil Kumar Singh and Indranil Saha. 2023. Stl-based synthesis of feedback controllers using reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.
[45] Adam Stooke, Joshua Achiam, and Pieter Abbeel. 2020. Responsive safety in reinforcement learning by pid lagrangian methods. In *International Conference on Machine Learning*. PMLR, 9133–9143.
[46] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. 2019. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation* (2019).
[47] Jianwen Sun, Tianwei Zhang, Xiaofei Xie, Lei Ma, Yan Zheng, Kangjie Chen, and Yang Liu. 2020. Stealthy and efficient adversarial attacks against deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.
[48] Yanchao Sun, Ruijie Zheng, Yongyuan Liang, and Furong Huang. 2021. Who is the strongest enemy? towards optimal and efficient evasion attacks in deep rl. *arXiv preprint arXiv:2106.05087* (2021).
[49] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. 2024. Gymnasium: A Standard Interface for Reinforcement Learning Environments. *arXiv preprint arXiv:2407.17032* (2024).
[50] Yixuan Wang, Simon Zhan, Zhilu Wang, Chao Huang, Zhaoran Wang, Zhuoran Yang, and Qi Zhu. 2023. Joint differentiable optimization and verification for certified reinforcement learning. In *Proceedings of the ACM/IEEE 14th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2023)*. 132–141.
[51] Yixuan Wang, Simon Sinong Zhan, Ruochen Jiao, Zhilu Wang, Wanxin Jin, Zhuoran Yang, Zhaoran Wang, Chao Huang, and Qi Zhu. 2023. Enforcing hard constraints with soft barriers: Safe reinforcement learning in unknown stochastic environments. In *International Conference on Machine Learning*. PMLR, 36593–36604.
[52] Yihang Yao, Zuxin Liu, Zhepeng Cen, Jiacheng Zhu, Wenhao Yu, Tingnan Zhang, and Ding Zhao. 2024. Constraint-conditioned policy optimization for versatile safe reinforcement learning. *Advances in Neural Information Processing Systems* 36 (2024).
[53] Haonan Yu, Wei Xu, and Haichao Zhang. 2022. Towards safe reinforcement learning with a safety editor policy. *Advances in Neural Information Processing Systems* 35 (2022), 2608–2621.
[54] Simon Sinong Zhan, Yixuan Wang, Qingyuan Wu, Ruochen Jiao, Chao Huang, and Qi Zhu. 2023. State-wise safe reinforcement learning with pixel observations. *arXiv preprint arXiv:2311.02227* (2023).
[55] Huan Zhang, Hongge Chen, Duane Boning, and Cho-Jui Hsieh. 2021. Robust reinforcement learning on state observations with learned optimal adversary. *arXiv preprint arXiv:2101.08452* (2021).

[56] Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Mingyan Liu, Duane Boning, and Cho-Jui Hsieh. 2020. Robust deep reinforcement learning against adversarial perturbations on state observations. *Advances in Neural Information Processing Systems* (2020).

[57] Lin Zhang, Xin Chen, Fanxin Kong, and Alvaro A Cardenas. 2020. Real-time attack-recovery for cyber-physical systems using linear approximations. In *IEEE Real-Time Systems Symposium*. IEEE.

[58] Lin Zhang, Mengyu Liu, and Fanxin Kong. 2023. AI-enabled Real-Time Sensor Attack Detection for Cyber-Physical Systems. In *AI Embedded Assurance for Cyber Systems*. Springer, 91–120.

[59] Yiming Zhang, Quan Vuong, and Keith Ross. 2020. First order constrained optimization in policy space. *Advances in Neural Information Processing Systems* 33 (2020), 15338–15349.

[60] Aditya Zutshi, Jyotirmoy V Deshmukh, Sriram Sankaranarayanan, and James Kapinski. 2014. Multiple shooting, cegar-based falsification for hybrid systems. In *Proceedings of the 14th International Conference on Embedded Software*. 1–10.