Vulnerability Exploration of Safe Reinforcement Learning in Cyber-Physical Systems via STL Mining

Jialiang Fan
jfan5@nd.edu
University of Notre Dame
Notre Dame, Indiana, USA

Shixiong Jiang
sjiang5@nd.edu
University of Notre Dame
Notre Dame, Indiana, USA

Mengyu Liu
mengyu.liu@wsu.edu
Washington State University
Tri-Cities, Washington, USA

Fanxin Kong
fkong@nd.edu
University of Notre Dame
Notre Dame, Indiana, USA

Abstract

Safe Reinforcement Learning (safe RL) has been widely used in safety-critical cyber-physical systems (CPS) to achieve task goals while satisfying safety constraints. Analyzing vulnerabilities that can be exploited to violate safety (i.e., safety-violated vulnerabilities) is crucial for understanding and improving the robustness of safe RL policies in CPS. However, existing works are inadequate for addressing such vulnerabilities, as they either focus on vulnerabilities that merely degrade task performance (rather than causing safety violations) or rely on strong assumptions about an adversary's capability (e.g., requiring explicit knowledge of the safety constraints). This paper aims to bridge this gap by studying safety-violated vulnerabilities of safe RL in CPS without requiring prior knowledge of the underlying safety constraints. To this end, we propose a novel adversarial framework based on Signal Temporal Logic (STL) mining. The framework first mines STL formulas to uncover the implicit safety constraints of a safe RL policy, and then synthesizes perturbation attacks that violate these constraints. The generated attacks can effectively and efficiently induce safety violations by adapting perturbations and identifying critical time intervals for applying them. We conduct extensive experiments across multiple CPS environments, and the results demonstrate the effectiveness and efficiency of our method.

CCS Concepts

• **Computer systems organization** → **Embedded and cyber-physical systems**; • **Computing methodologies** → **Reinforcement learning**; • **Security and privacy** → *Logic and verification.*

Keywords

Cyber-Physical Systems, Safe Reinforcement Learning, Signal Temporal Logic Mining, Adversarial Safety-Violation Attack

ACM Reference Format:

Jialiang Fan, Shixiong Jiang, Mengyu Liu, and Fanxin Kong. 2026. Vulnerability Exploration of Safe Reinforcement Learning in Cyber-Physical Systems via STL Mining. In *Proceedings of 29th ACM International Conference on Hybrid Systems: Computation and Control, and 17th ACM/IEEE International Conference on Cyber-Physical Systems (HSCC/ICCPS '26).* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

HSCC/ICCPS '26, Saint Malo, France
2026. ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Safe Reinforcement Learning (safe RL), aiming to achieve task goals while satisfying safety constraints, has been widely applied to safety-critical domains such as robotic systems [10] and various cyber-physical systems [19, 20]. Safe RL policies usually employ neural networks and thus may inherently exhibit vulnerabilities when faced with unexpected or adversarial conditions. Exploiting such vulnerabilities can cause safety violations leading to serious consequences in the real world. Hence, studying the robustness of safe RL policies in an adversarial setting is essential for their deployment in practice.

Existing works on adversarial RL are confined to degrading task reward (and thus the performance), rather than safety violations, such as [7, 28, 34]. However, in the context of safe RL, safety satisfaction is critical as such violations can cause serious consequences in the real world. Note that task reward reduction is not equivalent to safety violation. That is, attacks that reduce task reward may not violate safety, while attacks that violate safety may even increase task reward [16, 21]. Hence, new studies with focus on vulnerability leading to safety violations (i.e. safety-violated vulnerability) are needed to understand and improve the robustness of safe RL.

There are several pioneer works on safety-violated vulnerability [15, 16, 21]. However, they impose strict assumptions on an adversary's capability. Liu et al. [21] require an adversary's direct access to the gradients of cost critic and reward critic networks. Jiang et al. [16] assume that an attacker knows explicit safety constraints. In addition, all the existing works apply perturbations at every time step as the safe RL policy runs. This not only implicitly assumes an adversary having such high capability, but also lowers attack efficiency as occasionally applying perturbations can be sufficient for safety violations. Therefore, effective and efficient adversarial strategies for safe RL is needed in a more practical setting.

To achieve this, we propose a novel adversarial framework to explore safety-violated vulnerability based on Signal Temporal Logic (STL) mining. STL is a formal language that can express safety, liveness, temporal and other properties of a task. To address the unavailability of explicit safety constraints, the framework first mines STL formulas over collected trajectories to characterize the safety constraints which are implicitly embedded in a safe RL policy. Then, the mined formulas are used to train surrogate and adversarial models if model parameters of the victim policy are inaccessible.

Finally, an adversarial strategy is synthesized to generate perturbation and decide the critical time to apply the perturbation for safety violations.

To be specific, our major technical contributions are summarized as follows.

(1) We propose an STL-mining approach that can accurately identify safety constraints of a safe RL policy.
(2) We propose an attack synthesis approach that can effectively and efficiently violate the safety of a safe RL policy.
(3) We carry out extensive experiments using multiple environments and the results show the superior performance of our method compared to existing ones.

The remainder of this paper is organized as follows. Section 2 reviews related work on adversarial attacks in reinforcement learning and STL mining. Section 3 introduces the necessary preliminaries on safe RL and STL mining and presents the threat model. Section 4 formalizes the problem, describes our proposed attack framework, and provides the theoretical analysis. Section 5 reports the experimental results. Section 6 discusses the limitations of the proposed approach and potential defense strategies. Finally, Section 7 concludes the paper.

2 Related Work

The related work is organized into two parts: adversarial attacks on RL and STL mining.

2.1 Adversarial Attacks on RL

Huang *et al.* [11] prove that FGSM also works on reinforcement learning algorithms. Subsequently, researchers have proposed many gradient-based methods to more stealthily and efficiently attack RL algorithms [7, 30]. Some studies also introduce robust training methods to defend against the attack [18, 32, 34]. However, these methods focus solely on the rewards in reinforcement learning (RL). Liu *et al.* [21] proposed max-reward and max-cost attacks for safe RL algorithms. Jiang *et al.* [16] introduced a vulnerability exploration method for safe RL as well. However, as discussed in the Introduction, these two prior works rely on policy gradients and safety information.

2.2 STL Mining

STL mining has been applied in various domains, including verification [6, 24], traffic monitoring [23, 26], and classification tasks [1, 5]. Jha *et al.* [12] proposed a gradient-based algorithm to determine the optimal parameters of parametric STL (pSTL) and develop an open-source tool, TeLEx for STL mining. Leung *et al.* [17] introduced a method that transforms STL robustness values into computational graphs, enabling the use of gradient-based optimization for robotic applications. An interesting work was done by Yifru *et al.* [33], which leverages Bayesian optimization to tune pSTL parameters in order to accelerate safe RL training.

3 Background and Threat Model

In this section, we introduce the background on safe RL and STL mining, and present the threat model for the proposed STL mining–guided adversarial attack.

3.1 Safe RL

The safe RL [27] problem is usually formulated as a constrained Markov Decision Process (CMDP) represented by $(\mathcal{S}, \mathcal{A}, r, c, p, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $r : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \to \mathbb{R}$ denotes the reward obtained by taking action a at state s, $c : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^+$ denotes the cost associated with violating safety constraints, $p : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ denotes the transition kernel from a state–action pair (s, a) to the next state s', and $\gamma \in [0, 1)$ is the discount factor.

The goal of safe RL is to maximize the expected return while ensuring that the expected cost remains below a predefined threshold d, which can be formally expressed as

$$\pi^* = \arg\max_{\pi} V_r^{\pi}, \quad \text{s.t.} \quad V_c^{\pi} \le d, \tag{1}$$

where

$$V_r^{\pi} = \mathbb{E}_{\pi}\Big[\sum_{t=0}^{\infty} \gamma^t r_t\Big], \quad V_c^{\pi} = \mathbb{E}_{\pi}\Big[\sum_{t=0}^{\infty} \gamma^t c_t\Big]$$

denote the discounted cumulative reward and cost under policy π, respectively. Although the CMDP is formulated with an infinite horizon, in practice all training and evaluation are performed over finite episodes of length T, which is standard in safe RL benchmarks [13]. Among various safe RL methods, a common approach is to extend traditional RL algorithms using the Lagrangian relaxation technique. Examples include PPO-Lagrangian [27] and PID-Lagrangian-TRPO [29].

3.2 STL Mining

Signal Temporal Logic (STL) [22] is a formal specification framework that extends temporal logic to continuous-time, real-valued signals, enabling the precise expression of properties and constraints over time-varying data. The syntax of STL is given by

$$\begin{aligned} \phi ::= \; & \mu \mid \text{true} \mid \neg\phi \mid \phi \vee \phi \mid \phi \wedge \phi \\ & \mid \mathbf{G}_{[a,b)}\phi \mid \mathbf{F}_{[a,b)}\phi \mid \phi_1 \mathbf{U}_{[a,b)}\phi_2, \end{aligned} \tag{2}$$

where $\mu ::= f(x_t) \sim d$ is an atomic proposition with $\sim \in \{<, \le, >, \ge\}$ and $d \in \mathbb{R}$ a threshold; x denotes a continuous-time, continuous-valued signal, and x_t is the value of x at time t; $[a, b)$ is a time interval; $\neg$ and $\wedge$ denote logical negation and conjunction; and $\mathbf{G}$, $\mathbf{F}$, and $\mathbf{U}$ are temporal operators representing "always", "eventually", and "until", respectively.

In addition to its Boolean semantics, STL provides quantitative semantics, denoted by $\rho(\tau, \phi, t)$ [8, 9], which assigns a real-valued robustness score to a temporal trajectory τ with respect to formula ϕ at time instant t. The robustness is defined recursively:

$$\begin{aligned} \rho(\tau, \mu, t) &= f(x_t) - d, \\ \rho(\tau, \neg\phi, t) &= -\rho(\tau, \phi, t), \\ \rho(\tau, \phi_1 \wedge \phi_2, t) &= \min\big(\rho(\tau, \phi_1, t),\ \rho(\tau, \phi_2, t)\big), \\ \rho(\tau, \phi_1 \vee \phi_2, t) &= \max\big(\rho(\tau, \phi_1, t),\ \rho(\tau, \phi_2, t)\big), \\ \rho(\tau, \mathbf{G}_{[a,b)}\phi, t) &= \min_{t' \in [t+a,\, t+b)} \rho(\tau, \phi, t'), \\ \rho(\tau, \mathbf{F}_{[a,b)}\phi, t) &= \max_{t' \in [t+a,\, t+b)} \rho(\tau, \phi, t'). \end{aligned} \tag{3}$$

A positive robustness value implies satisfaction, a negative value implies violation, and the absolute value indicates the distance to the satisfaction boundary.

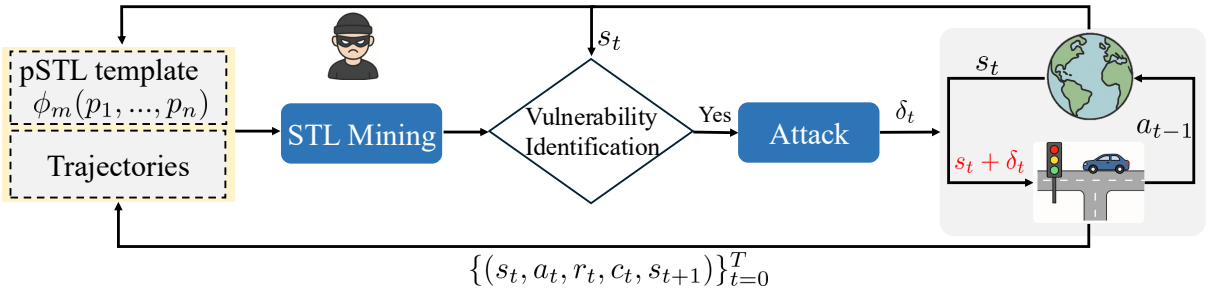


Figure 1: Diagram of the STLMAA for safe RL based CPS. The safety constraints are mined as an STL formula from both collected trajectories and attacker-observed pSTL formulas. Then, the STLMAA identifies the vulnerable state and generates adversarial perturbations applied to the system's observation, ultimately causing safety violation.

Motivated by the utility of quantitative semantics, STL mining [3] aims to automatically extract temporal logic specifications from trajectory data, thereby supporting system understanding, maintenance, and reverse engineering. A common formulation relies on parametric STL (pSTL), where we express a pSTL template as $\phi(p)$, with $p = [p_1, p_2, \ldots, p_n]^\top$ denoting the parameter vector to be optimized. These parameters typically encode predicate thresholds and temporal bounds (e.g., safety margins or timing requirements). Given system trajectories, the objective is to find an optimal parameter vector $p_m = [p_{1m}, p_{2m}, \ldots, p_{nm}]^\top$ such that the resulting formula $\phi(p_m)$ aligns with observed behaviors and captures the underlying safety conditions.

Among existing approaches, template-based STL mining is one of the most widely used [2, 4, 12]. It starts by specifying a pSTL template whose parameters determine temporal intervals or predicate thresholds. The mining problem is then cast as an optimization task, in which quantitative semantics provide a continuous measure for evaluating candidate formulas, and optimization techniques are applied to search for the parameters that best fit the trajectory data.

3.3 Threat Model

Attack Scenario. We consider an attacker that interferes with a safe RL agent during its inference phase. The agent, modeled as a CMDP, operates in a real or simulated environment where sensory observations may be subject to noise, faults, or malicious interference. The attacker perturbs the observation stream by injecting crafted noise into the sensor inputs, thereby misleading the control policy without directly manipulating the environment or the agent's internal model.

Attack Capability. The attacker can passively observe the system's behavior during inference and collect trajectories, and can actively inject additive perturbations into the agent's observations. At each time step t, the attacker applies a perturbation δ_t to the observation, yielding a perturbed input. To preserve stealthiness, the perturbation is bounded by a fixed budget $\kappa > 0$, e.g.,

$$\|\delta_t\|_\infty \leq \kappa,$$

which follows common settings adopted in prior work on adversarial attacks against RL [16].

Adversary Knowledge and Attack Levels. The attacker is assumed to have access to system trajectories, each of which is denoted as

$$\tau = \{(s_t, a_t, r_t, c_t, \ldots, s_{t+1})\}_{t=0}^{T}, \tag{4}$$

where at each discrete time step t, s_t is the system state, a_t is the action taken by the agent, r_t is the received reward, and c_t is the incurred cost. The attacker is assumed to possess only minimal prior knowledge about the system, such as the general task type (e.g., navigation, velocity regulation, or obstacle avoidance) and the physical meaning of a few key state variables. This prior knowledge is used to manually construct a pSTL template $\phi_m(\mathbf{p})$ that qualitatively reflects the expected safety requirement (e.g., bounds on position or velocity), while the concrete parameter values $\mathbf{p}$ are subsequently identified from data via STL mining.

Depending on the specific knowledge available to the attacker, we distinguish three high-level attack types based on access to the control policy π and transition dynamics f. As summarized in Table 1, the gray-box case is further split into two sublevels depending on whether the attacker knows π or f, which correspond to the gray-box-π and gray-box-f settings used later in our attack design and experiments.

Table 1: Adversary knowledge assumptions under different attack levels.

Attack Type	π	f
White-box	✓	✓
Gray-box (π)	✓	×
Gray-box (f)	×	✓
Black-box	×	×

4 STL Mining Guided Adversarial Attack

In this section, we first formalize the problem of STL mining–guided adversarial attack (STLMAA), then present the STLMAA framework, and finally provide theoretical robustness deviation analyses.

4.1 Problem Definition

Recall that $\rho(\tau, \phi, t)$ denotes the robustness of trajectory τ with respect to an STL formula ϕ at time step t under the quantitative semantics. To assess how well the mined STL constraints capture the true safety properties, we first define mining effectiveness.

Definition 1 (Mining Effectiveness). Consider a parametric STL template $\phi(p)$ with parameter vector $p \in \mathbb{R}^k$, and let $p^* \in \mathbb{R}^k$ denote the (unknown) ground-truth parameters representing the true safety constraints. Let p_m be the parameter vector returned by an STL mining algorithm, and $\phi_m \coloneqq \phi(p_m)$ the corresponding mined formula. We say that the mining is *effective* if

$$\forall \tau \in \mathcal{D}_{\text{safe}},\ \forall t,\ \rho(\tau, \phi_m, t) > 0,$$

and

$$\|p_m - p^*\| \le \epsilon,$$

for some small $\epsilon > 0$, where $\mathcal{D}_{\text{safe}}$ denotes the set of trajectories that satisfy the true safety constraints.

Definition 1 formalizes both a semantic condition (all observed safe trajectories satisfy the mined formula with positive robustness) and a parameter-level approximation guarantee. Intuitively, ϕ_m provides a tight and faithful approximation of the unknown ground-truth safety specification, enabling it to serve as a reliable guide for adversarial attack generation.

Definition 2 (Attack Effectiveness). Consider a safe RL system and a mined safety specification ϕ_m. For a given start time t_0 and finite horizon $T > 0$, an adversarial attack is said to be *effective* if it induces a violation of the STL constraint over the horizon, i.e.,

$$\rho\big(\tau, \mathbf{G}_{[t_0, t_0+T]}\phi_m, t_0\big) < 0. \tag{5}$$

In the CMDP view, such violations are typically reflected by increased cost signals (e.g., going out of bounds or collisions).

Definition 2 links safety violations in safe RL systems directly to STL robustness: a negative robustness value corresponds to a violation of the mined safety constraint ϕ_m. This allows us to use ρ both as a safety metric and as an optimization objective for attack design. Since robustness is defined with respect to ϕ_m, the effectiveness of an attack inherently depends on the mining effectiveness in Definition 1.

Definition 3 (Attack Efficiency). Given a fixed perturbation budget $\kappa > 0$, let $C^{(j)}$ denote the cumulative cost of the j-th perturbed episode under a given attack strategy, and let M be the total number of evaluation episodes. The *attack efficiency* (AE) is defined as

$$\text{AE} \ = \ \frac{1}{M} \sum_{j=1}^{M} \frac{C^{(j)}}{\kappa}, \tag{6}$$

i.e., the average episode cost per unit attack power. A higher AE indicates that the attack causes more severe safety violations per unit perturbation magnitude, reflecting a more efficient (and potentially more stealthy) attack.

Definition 3 provides a quantitative metric to compare different attack strategies under the same perturbation budget: for a fixed κ, a more efficient attack achieves larger cost (stronger violations) with the same perturbation strength.

We now formulate the attack problem. Our goal is to design a perturbation mechanism that leverages the mined safety specification ϕ_m to systematically degrade the agent's adherence to safety constraints. Concretely, at each time step t, the attacker injects an additive perturbation δ_t into the current observation,

$$s'_t = s_t + \delta_t,$$

with $\|\delta_t\|_\infty \le \kappa$. The perturbed observation is processed by the policy to yield a (potentially misguided) action

$$a'_t = \pi(s'_t), \tag{7}$$

which is then applied to the system dynamics to produce the next perturbed state

$$s'_{t+1} = f(s'_t, a'_t). \tag{8}$$

The trajectory τ is thus extended with perturbed transitions, and the robustness of ϕ_m can be re-evaluated at subsequent time steps. Since safe RL systems are typically modeled as Markov decision processes, the future evolution depends only on the current perturbed state. As a result, a greedy one-step attack—that minimizes the robustness at the next step—can remain effective in multi-step scenarios, as each local perturbation propagates through future transitions.

To guide the system toward violating the safety constraints, we formulate the attack objective as minimizing the robustness at the next observation step:

$$\min_{\delta_t}\ \rho(\tau, \phi_m, t{+}1).$$

Taking into account the perturbation and dynamics constraints, we arrive at the following constrained optimization problem:

$$\begin{aligned} \delta_t \ = \ & \arg\min_{\delta_t}\ \rho(\tau, \phi_m, t{+}1) \\ \text{s.t.}\ \ & \|\delta_t\|_\infty \le \kappa, \\ & s'_t = s_t + \delta_t, \\ & a'_t = \pi(s'_t), \\ & s'_{t+1} = f(s'_t, a'_t). \end{aligned} \tag{9}$$

This formulation characterizes the adversarial attack as a robustness minimization problem under bounded perturbations. By iteratively applying such perturbations along an episode, the adversary seeks to induce violations of the mined safety specification ϕ_m, thereby exposing vulnerabilities in the safe RL agent.

4.2 Attack Framework

We now present the STLMAA framework. An overview is shown in Fig. 1. The framework consists of three main components: STL mining, vulnerability identification, and attack generation.

4.2.1 STL Mining. The first step of our framework is to extract STL safety constraints from the trajectories collected from the victim safe RL system. As discussed in Section 3.3, we assume that the attacker can interact with the system during inference and obtain state–action–cost trajectories. Using the cost signal c_t, we remove trajectories that contain explicit safety violations, resulting in a dataset that reflects nominally safe system behavior.

Given basic prior knowledge about the task and the meaning of key state variables (e.g., navigation tasks with position bounds or velocity limits), the attacker constructs a parametric STL (pSTL) template $\phi(p)$, where $p = [p_1, \ldots, p_n]^\top$ denotes the set of parameters to be mined. Following standard template-based STL mining [2, 4, 12], the logical structure of the template is manually specified based on this high-level prior knowledge, while the parameter values p are automatically inferred from the collected trajectories via quantitative robustness optimization.

Algorithm 1 STL Mining

Require: pSTL template $\phi(p)$; trajectories $\{\tau_i\}$; iteration limit K; initial parameters $p^{(0)}$; step size η.
Ensure: Mined parameters p_m and corresponding STL formula $\phi_m = \phi(p_m)$.

for $k = 0$ to $K-1$ **do**
  Compute robustness $\rho_i^{(k)} = \rho(\tau_i, \phi(p^{(k)}), t_0)$ for all i
  $C^{(k)} \leftarrow \{\, i \mid \rho_i^{(k)} = \min_j \rho_j^{(k)} \,\}$ ▷ critical trajectories
  if $\min_i \rho_i^{(k)} \le 0$ **then**
    Compute gradient $\nabla_p^{(k)}$ using limiting predicates on $C^{(k)}$
    $p^{(k+1)} \leftarrow p^{(k)} + \eta\, \nabla_p^{(k)}$ ▷ restore feasibility
  else
    Identify predicates with largest robustness margin
    Compute tightening direction $\nabla_p^{(k)}$
    $p^{(k+1)} \leftarrow p^{(k)} - \eta\, \nabla_p^{(k)}$ ▷ shrink slack
  end if
  if $\min_i \rho_i^{(k)} < \epsilon$ **then**
    break
  end if
end for
$p_m \leftarrow p^{(k+1)}$, $\phi_m \leftarrow \phi(p_m)$
return p_m, ϕ_m

Algorithm 1 summarizes a generic template-based STL mining procedure. Given $\phi(p)$ and trajectories $\{\tau_i\}$, the algorithm iteratively searches for a parameter vector p that yields positive robustness while being as tight as possible. At each iteration k, the robustness $\rho_i^{(k)}$ of each trajectory under $\phi(p^{(k)})$ is computed, and the set of critical trajectories $C^{(k)}$ with minimal robustness is identified. If any trajectory exhibits non-positive robustness, a feasibility-restoring update is taken to increase robustness on the limiting predicates. Otherwise, a tightening step shrinks the slack by reducing the robustness margin of the loosest predicates. The process terminates when the minimum robustness across trajectories falls below a small threshold $\epsilon > 0$ or the maximum number of iterations is reached, yielding an ϵ-tight STL formula $\phi_m = \phi(p_m)$.

4.2.2 Vulnerability Identification. Given the mined STL formula ϕ_m, we next identify critical states where attacks are more likely to succeed. For each state s_t along a trajectory, we compute its robustness with respect to ϕ_m as $\rho(\tau, \phi_m, t)$. Smaller robustness (especially near zero) indicates proximity to a potential safety violation.

To adapt to the evolving behavior of the system, we employ a sliding window of recent robustness values and estimate a dynamic vulnerability threshold. Specifically, we maintain a buffer of the most recent W robustness values, and once the buffer is full, we update a threshold θ using a percentile-based rule:

$$\theta = \text{Percentile}(\text{history}, p_v),$$

where p_v is a chosen percentile. In our experiments, we set $W = 20$ and $p_v = 10\%$, meaning that when the current robustness falls below the 10th percentile of the recent history, the corresponding time step is considered *vulnerable*.

Algorithm 2 Attack Generation (Generic Skeleton)

Require: Safe RL policy (or surrogate) π; current state s_t; perturbation bound κ; number of iterations n; action bounds a^+, a^-; target action $\bar{a}_t$; distance function $l(\cdot,\cdot)$; step size α.
Ensure: Perturbation δ_t.

$s'_t \leftarrow s_t$ ▷ initialize perturbed state
for $i = 1$ to n **do**
  $a'_t \leftarrow \pi(s'_t)$
  $a'_t \leftarrow \max\big(\min(a'_t, a^+), a^-\big)$ ▷ enforce action bounds
  state_grad $\leftarrow \nabla_{s'_t}\, l(a'_t, \bar{a}_t)$
  $s'_t \leftarrow s'_t - \alpha\, \kappa\, \text{sign}(\text{state\_grad})$
  $s'_t \leftarrow \max\big(\min(s'_t, s_t + \kappa), s_t - \kappa\big)$ ▷ clip to perturbation budget
end for
$\delta_t \leftarrow s'_t - s_t$
return δ_t

In practice, we implement this mechanism via a threshold updater class (e.g., `STLThresholdUpdater`) that:

- applies a *cold start*: no threshold is computed until the history buffer reaches size W, to avoid unstable thresholds;
- updates the vulnerability threshold θ using percentile statistics over the history;
- integrates with an attack decision function (`check_attack`) that triggers the attack logic whenever $\rho(\tau, \phi_m, t) \le \theta$.

In summary, the mined STL formula ϕ_m is used to compute robustness values in real time, and the dynamic thresholding mechanism identifies vulnerable moments in the trajectory. This allows STLMAA to focus perturbations on states that are already close to the safety boundary, thereby improving attack efficiency.

4.2.3 Attack Generation. After identifying vulnerable states, STLMAA generates adversarial perturbations to be applied selectively at those time steps. Depending on the adversary's knowledge (cf. Section 3.3), we design different attack strategies. Algorithm 2 provides a generic attack generation skeleton, where the specific instantiation of the target action $\bar{a}_t$ and the policy π depends on the attack level.

White-box Attack. When **both** the transition function f and the control policy π are available:

(1) The attacker uses f and ϕ_m to compute a target action $\bar{a}_t$ that approximately minimizes the next-step robustness (e.g., by solving a one-step optimization over a_t).
(2) The true policy π is used inside Algorithm 2 to obtain $a'_t = \pi(s'_t)$ and its gradients with respect to s'_t.

This yields a fully white-box, gradient-based attack that directly exploits knowledge of both policy and dynamics.

Gray-box Attack. In the gray-box setting, the attacker knows **either** f or π but not both. We further distinguish two subtypes, consistent with Table 1:

- **Gray-box-π attack (known π, unknown f).** When the policy π is known but the transition function f is not, computing $\bar{a}_t$ by explicit robustness minimization over the next state becomes difficult. To address this, we train

an adversarial RL agent with policy π_{adv} whose reward is defined using the negated safety specification $\phi_{\text{adv}} = \neg\phi_m$, encouraging trajectories that violate the mined safety constraint. The target action in Algorithm 2 is then set to

$$\bar{a}_t = \pi_{\text{adv}}(s_t),$$

while the true victim policy π is used in Line 4 for $a'_t = \pi(s'_t)$.

- **Gray-box-f attack (known f, unknown π).** When the transition function f is known but the victim policy π is unknown, the attacker cannot directly compute gradients through π. In this case, we train a surrogate policy π_{sur} from collected trajectories to imitate the victim's behavior. The surrogate is then used in Algorithm 2 by replacing $a'_t = \pi(s'_t)$ with

$$a'_t = \pi_{\text{sur}}(s'_t),$$

while $\bar{a}_t$ can still be obtained using f and ϕ_m (or via π_{adv} if desired).

Black-box Attack. When the attacker has **no direct access** to either f or π, we combine both surrogate modeling and adversarial learning:

- A surrogate policy π_{sur} is trained from observed trajectories to approximate the victim's action mapping.
- An adversarial RL policy π_{adv} is trained with reward derived from $\neg\phi_m$ to produce actions that tend to violate the safety constraints.
- In Algorithm 2, we set

$$\bar{a}_t = \pi_{\text{adv}}(s_t), \qquad a'_t = \pi_{\text{sur}}(s'_t).$$

This enables a fully black-box attack that only relies on observable trajectories and robustness feedback, without access to the true dynamics or policy internals.

4.3 Theoretical Analyses

We now analyze how STLMAA-induced perturbations affect STL robustness, and how mining errors propagate into the overall robustness deviation. We first derive one-step bounds on action and state deviations, then relate these to the robustness of a fixed STL formula, and finally incorporate the additional error introduced by STL mining.

Standing assumptions. Throughout this subsection we assume:

(1) The safe RL policy $\pi : \mathcal{S} \to \mathcal{A}$ is L-Lipschitz, i.e.,

$$\|\pi(s) - \pi(s')\| \le L \|s - s'\|, \quad \forall s, s' \in \mathcal{S}.$$

(2) The system dynamics $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ are (locally) Lipschitz in the control input with constant L_s, i.e.,

$$\|f(s,a) - f(s,\tilde{a})\| \le L_s \|a - \tilde{a}\|, \quad \forall s \in \mathcal{S},\ a, \tilde{a} \in \mathcal{A}.$$

(3) For a fixed STL formula ϕ, the robustness of ϕ evaluated at a single state s, denoted by $\rho(\phi, s)$, is Lipschitz on $\mathcal{S}$ with constant $K_\rho > 0$:

$$|\rho(\phi, s) - \rho(\phi, s')| \le K_\rho \|s - s'\|, \quad \forall s, s' \in \mathcal{S}.$$

For the one-step analysis, we consider the instantaneous robustness

$$\rho(\tau, \phi, t) \coloneqq \rho(\phi, s_t).$$

(4) For a parameterized STL template $\phi(p)$ with $p \in \mathbb{R}^k$, the robustness $\rho(\tau, \phi(p), t)$ is Lipschitz in p uniformly over trajectories τ and time indices t, with constant $L_p > 0$:

$$\left|\rho(\tau, \phi(p), t) - \rho(\tau, \phi(p'), t)\right| \le L_p \|p - p'\|, \quad \forall p, p' \in \mathbb{R}^k.$$

Lemma 1 (Adversarial Action Bound). *Suppose the safe RL policy π is L-Lipschitz continuous over $\mathcal{S}$. Let $s' \in \mathcal{S}$ be an adversarially perturbed version of s satisfying $\|s' - s\| \le \delta$, and define $a = \pi(s)$ and $a' = \pi(s')$. Then*

$$\|a' - a\| \le L\,\delta.$$

Proof. By L-Lipschitz continuity of π,

$$\|\pi(s') - \pi(s)\| \le L \|s' - s\|.$$

Since $\|s' - s\| \le \delta$ and $a = \pi(s)$, $a' = \pi(s')$, we obtain

$$\|a' - a\| = \|\pi(s') - \pi(s)\| \le L\,\delta.$$

□

Lemma 1 bounds how much the policy output can change in response to bounded state perturbations.

Lemma 2 (One-step State Deviation Bound). *Under the assumptions above, let the system dynamics be*

$$s_{n+1} = f(s_n, a_n), \qquad s'_{n+1} = f(s_n, a'_n),$$

where $a_n = \pi(s_n)$ and $a'_n = \pi(s'_n)$, and $\|s'_n - s_n\| \le \delta$. If f is Lipschitz in its second argument with constant L_s and $\|a'_n - a_n\| \le L\,\delta$ (Lemma 1), then

$$\|s_{n+1} - s'_{n+1}\| \le L_s \|a_n - a'_n\| \le L_s L\,\delta.$$

Proof. From the definitions of s_{n+1} and s'_{n+1},

$$s_{n+1} - s'_{n+1} = f(s_n, a_n) - f(s_n, a'_n).$$

Taking norms and using Lipschitz continuity of f in a,

$$\|s_{n+1} - s'_{n+1}\| = \|f(s_n, a_n) - f(s_n, a'_n)\| \le L_s \|a_n - a'_n\| \le L_s L\,\delta,$$

where the last inequality uses Lemma 1. □

Lemma 2 shows how the next-state deviation scales with both the policy smoothness and the input–state sensitivity of the dynamics.

Theorem 1 (One-step STL Robustness Difference Bound). *Under the assumptions in Lemma 1 and Lemma 2, let ϕ be an STL formula whose instantaneous robustness is*

$$\rho(\tau, \phi, t) \coloneqq \rho(\phi, s_t),$$

where $\rho(\phi, \cdot)$ is K_ρ-Lipschitz on $\mathcal{S}$. Then, for the original trajectory τ and the perturbed trajectory τ',

$$|\rho(\tau, \phi, t{+}1) - \rho(\tau', \phi, t{+}1)| \le K_\rho\, L_s L\,\delta.$$

Proof. By definition,

$$\rho(\tau, \phi, t{+}1) = \rho(\phi, s_{t+1}), \qquad \rho(\tau', \phi, t{+}1) = \rho(\phi, s'_{t+1}).$$

Using Lipschitz continuity of $\rho(\phi, \cdot)$,

$$\begin{aligned}|\rho(\tau, \phi, t{+}1) - \rho(\tau', \phi, t{+}1)| &= |\rho(\phi, s_{t+1}) - \rho(\phi, s'_{t+1})| \\ &\le K_\rho \|s_{t+1} - s'_{t+1}\|.\end{aligned}$$

By Lemma 2, $\|s_{t+1} - s'_{t+1}\| \le L_s L\,\delta$, hence

$$|\rho(\tau, \phi, t{+}1) - \rho(\tau', \phi, t{+}1)| \le K_\rho\, L_s L\,\delta.$$

□

Theorem 1 thus bounds the one-step change in robustness induced by a bounded perturbation of the current state.

Lemma 3 (Robustness Bound Induced by ϵ-Tight STL Mining). *Let $\phi(p)$ be a parameterized STL formula with parameter vector $p \in \mathbb{R}^k$, and let p^* be the ground-truth parameters. Suppose the mined parameters p_m satisfy $\|p_m - p^*\| \le \epsilon$ for some $\epsilon > 0$. Assume that $\rho(\tau, \phi(p), t)$ is Lipschitz in p uniformly over trajectories τ and times t, with constant $L_p > 0$, i.e.,*

$$\left|\rho(\tau, \phi(p), t) - \rho(\tau, \phi(p'), t)\right| \le L_p \|p - p'\|, \quad \forall p, p' \in \mathbb{R}^k, \ \forall \tau, t.$$

Then, for any τ and t,

$$\left|\rho(\tau, \phi(p_m), t) - \rho(\tau, \phi(p^*), t)\right| \le L_p \, \epsilon.$$

Proof. Since $\|p_m - p^*\| \le \epsilon$ and $\rho(\tau, \phi(p), t)$ is L_p-Lipschitz in p,

$$\left|\rho(\tau, \phi(p_m), t) - \rho(\tau, \phi(p^*), t)\right| \le L_p \|p_m - p^*\| \le L_p \, \epsilon.$$

□

Theorem 2 (Bounded Robustness Deviation in STLMAA). *Let ϕ be the original STL formula and $\phi_m = \phi(p_m)$ the mined formula with $\|p_m - p^*\| \le \epsilon$. Let s_{n+1} and s'_{n+1} be the unperturbed and perturbed system states at time $n+1$, and let τ_{n+1} and τ'_{n+1} denote the trajectories initialized at s_{n+1} and s'_{n+1}, respectively. Define the sensitivity constant*

$$K_\phi = \sup_{s, s' \in \mathcal{S}} \frac{|\rho(\tau(s), \phi, 0) - \rho(\tau(s'), \phi, 0)|}{\|s - s'\|},$$

where $\tau(s)$ and $\tau(s')$ are trajectories initialized at s and s'. Assume that $\|s_{n+1} - s'_{n+1}\| \le L_s L \, \delta$ (Lemma 2). Then

$$\left|\rho(\tau'_{n+1}, \phi_m, 0) - \rho(\tau_{n+1}, \phi, 0)\right| \le K_\phi L_s L \, \delta + 2 L_p \, \epsilon.$$

Proof. By Lemma 3,

$$|\rho(\tau_{n+1}, \phi_m, 0) - \rho(\tau_{n+1}, \phi, 0)| \le L_p \epsilon,$$

and

$$|\rho(\tau'_{n+1}, \phi_m, 0) - \rho(\tau'_{n+1}, \phi, 0)| \le L_p \epsilon.$$

By the definition of K_ϕ and $\|s_{n+1} - s'_{n+1}\| \le L_s L \, \delta$,

$$|\rho(\tau_{n+1}, \phi, 0) - \rho(\tau'_{n+1}, \phi, 0)| \le K_\phi \|s_{n+1} - s'_{n+1}\| \le K_\phi L_s L \, \delta.$$

Applying the triangle inequality,

$$\begin{aligned}
&\left|\rho(\tau'_{n+1}, \phi_m, 0) - \rho(\tau_{n+1}, \phi, 0)\right| \\
&\le \left|\rho(\tau_{n+1}, \phi_m, 0) - \rho(\tau_{n+1}, \phi, 0)\right| + \left|\rho(\tau_{n+1}, \phi, 0) - \rho(\tau'_{n+1}, \phi, 0)\right| \\
&\qquad + \left|\rho(\tau'_{n+1}, \phi, 0) - \rho(\tau'_{n+1}, \phi_m, 0)\right| \\
&\le L_p \epsilon + K_\phi L_s L \, \delta + L_p \epsilon \\
&= 2 L_p \, \epsilon + K_\phi L_s L \, \delta.
\end{aligned}$$

□

Theorem 2 shows that the total robustness deviation induced by STLMAA is provably bounded. The term $K_\phi L_s L \, \delta$ quantifies the sensitivity of the original STL specification ϕ to adversarial perturbations through the closed-loop dynamics, while $2L_p \, \epsilon$ captures the approximation error arising from using the mined formula ϕ_m instead of the ground-truth ϕ. Consequently, even when the attacker exploits imperfections in the mined STL constraints, the induced degradation of system safety remains within an analytically predictable margin.

5 Experiments

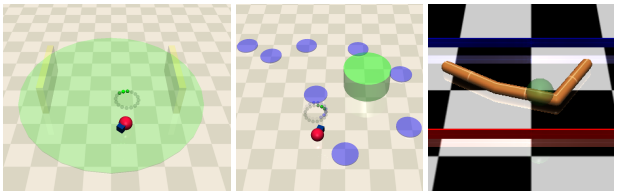
In this section, we evaluate the performance of STLMAA on three safe RL environments. All environments are implemented using the Safety-Gymnasium framework [13], which builds upon Safety-Gym and MuJoCo as simulators [31]. We first describe the environments, baselines, and training configurations, and then present the STL mining results and the attack performance of STLMAA.

5.1 Environments

Safe PointCircle Navigation. The Safe PointCircle environment evaluates an agent's ability to navigate within a circular region (radius 1.5 m) while avoiding two vertical wall constraints at $x = \pm 1$ m. The agent is encouraged to stay close to the circle boundary without violating safety constraints induced by the walls. The action space is 2D and bounded in $(-1, 1)$.

Safe PointGoal Navigation. The Safe PointGoal environment (Fig. 2(b)) evaluates an agent's ability to reach a designated goal while avoiding obstacles. The environment contains eight circular obstacles and one cylindrical goal. The agent must reach the goal within a limited horizon without collisions.

Safe SwimmerVelocity. We modify the original Safe SwimmerVelocity environment (Fig. 2(c)) so that it includes both position-level and velocity-level safety constraints. The task is to move as quickly as possible while satisfying $v_x < 0.2282$ m/s and $y \in [-1, 1]$ m, where v_x is the forward velocity and y is the lateral position.



(a) (b) (c)

Figure 2: (a) Safe PointCircle: navigation area (green) with two wall constraints. (b) Safe PointGoal: goal (green cylinder) with 8 obstacle circles (purple). (c) Safe SwimmerVelocity: environment with position and velocity constraints.

5.2 Baseline Adversarial Attackers

To comprehensively evaluate the effectiveness and efficiency of STLMAA, we compare it against several adversarial attackers commonly used in safe RL.

Random attacker. The random attacker adds i.i.d. perturbations sampled from a uniform distribution within a bounded ℓ_∞ ball to the observations at each step.

Gradient-based attacker. The gradient-based attacker follows the FGSM-style update [25]. Starting from the original state s, it iteratively constructs adversarial states s_i via

$$s_{i+1} = \Pi_{\mathcal{B}_\kappa(s)}\left(s_i + \alpha \cdot \mathrm{sign}(\nabla_s J(s_i, \pi^{\text{target}}))\right),$$

where J is an attack objective (e.g., negative return or cost), π^{target} is the victim policy, α is a step size, $\mathcal{B}_\kappa(s)$ is the perturbation ball

of radius κ around s, and $\Pi_{\mathcal{B}_\kappa(s)}$ denotes projection back onto the feasible perturbation set.

Max-cost attacker. The Max-cost attacker [21] optimizes perturbations by maximizing the cost critic provided by the safe RL algorithm. Let $q_c(s, a)$ denote the learned cost critic. The attacker performs gradient ascent on the loss

$$\mathcal{L}_{\text{max-cost}}(s) = -\|q_c(s, \pi(s))\|,$$

which encourages perturbations that increase the expected cumulative cost.

In addition to the baseline attackers (Random, Gradient-based, and Max-cost), we evaluate STLMAA under the four attacker knowledge levels introduced earlier (Table 1): white-box, gray-box-π, gray-box-f, and black-box. These settings correspond to the respective columns in Table 2 and the curves in Fig. 3.

5.3 Training Details

All environments are implemented using the Safety-Gymnasium framework [13], and we adopt OmniSafe [14] as our implementation of safe RL algorithms such as TD3-Lagrangian and PPO-Lagrangian.

5.3.1 Victim Policies.

Safe PointCircle Navigation. We train the victim policy using the TD3-Lagrangian algorithm [27] with a cost limit of 25. Training is performed for 1,000,000 environment steps (1,000 episodes, each with 1,000 steps). The resulting policy achieves a 96% success rate without attacks over 100 evaluation episodes, with an average reward of 66.79 and an average episode cost of 1.30.

Safe PointGoal Navigation. We train the victim policy using TD3-Lagrangian for 1,500 episodes, each with 1,000 time steps. We set the cost limit to 0 to enforce strict safety and introduce a collision buffer of 0.05 m to mimic real-world safety margins: when the Euclidean distance between the agent and any obstacle is less than 0.05 m, the instantaneous cost is set to 1, and 0 otherwise. The trained policy achieves a safety rate of 41% with an average reward of 22.62 and an average episode cost of 9.19 over 100 evaluation episodes.

Safe SwimmerVelocity. We train the victim policy using the PPO-Lagrangian algorithm for 1,000,000 environment steps (1,000 episodes, each with 1,000 steps). The resulting policy attains a safety rate of 72%, with an average reward of 18.58 and an average episode cost of 3.67 over 100 evaluation episodes.

5.3.2 Surrogate Model. For surrogate-model training, we employ the mined STL formula $\phi_m(\mathbf{p})$ as the safety constraint to compute the cost. Specifically, when the robustness value is negative (i.e., $\rho(\tau, \phi_m, t) < 0$), the cost is set to 1; otherwise, it is set to 0. The surrogate model is trained using safe RL to mimic the closed-loop behavior of the original system under the mined STL constraints.

Safe PointCircle Navigation. We train a surrogate policy using PPO-Lagrangian in OmniSafe for 1,000,000 steps over 1,000 epochs. Evaluating over 100 episodes yields an average reward of 55.67, an average episode cost of 0.40, and an average episode length of 1,000 steps.

Safe PointGoal Navigation. We train a surrogate policy using PPO-Lagrangian for 1,000,000 steps over 1,000 epochs. Over 100 evaluation episodes, the surrogate achieves an average reward of 7.13, an average episode cost of 27.45, and an average episode length of 1,000 steps.

Safe Swimmer Velocity. We train a surrogate policy using TD3-Lagrangian in OmniSafe for 1,000,000 steps over 1,000 epochs. Evaluating over 100 episodes yields an average reward of 25.36, an average episode cost of 161.85, and an average episode length of 1,000 steps.

5.3.3 Adversarial Model. To train the adversarial model, we modify the reward function of the Safety-Gymnasium environments so that lower STL robustness corresponds to higher reward for the adversary. This shaping encourages the adversarial policy to generate actions that drive the system towards safety violations.

Safe PointCircle Navigation. We train the adversarial model using PPO in OmniSafe for 1,000,000 steps over 1,000 epochs. Evaluated over 100 episodes, the learned adversary obtains an average episode reward of 5,090.50, an average episode cost of 860.60, and an average episode length of 1,000 steps.

Safe PointGoal Navigation. For the PointGoal environment, we reuse the same training schedule (PPO, 1,000,000 steps). Over 100 evaluation episodes, the adversarial model achieves an average episode reward of 0.199, an average episode cost of 15.40, and an average episode length of 1,000 steps.

Safe Swimmer Velocity. We train the adversarial model for Safe SwimmerVelocity using TD3 with 1,000,000 steps over 1,000 epochs. Evaluated on 100 episodes, the adversarial model achieves an average episode reward of 476.33, an average episode cost of 819.90, and an average episode length of 1,000 steps.

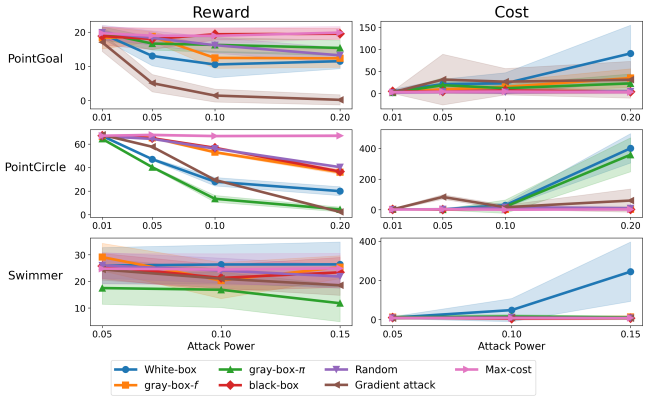


Figure 3: Comparison of rewards and costs for different attack methods across the three environments.

5.4 Safety Constraint Mining Results

Safe PointCircle Navigation. By inspecting the environment, we can intuitively construct a parametric STL (pSTL) template capturing the wall constraints:

$$\phi(x_u, x_l) = \mathbf{G}_{[0,T)}\big((x < x_u) \wedge (x > x_l)\big), \tag{10}$$

where T is the episode horizon, and x_u and x_l denote the upper and lower safety bounds on the agent's x-position, respectively. We

Table 2: Attack efficiency of different attack methods across environments.

Environment	Attack Power	White-box	Gray-box-f	Gray-box-π	Black-box	Random	Gradient	Max-cost
PointCircle	0.01	0.01	0.01	**0.01**	0.01	0.00	0.00	0.00
	0.05	0.01	0.01	0.00	0.00	0.00	**0.08**	0.00
	0.10	**0.05**	0.01	0.03	0.00	0.02	0.02	0.00
	0.20	**0.57**	0.01	0.50	0.01	0.01	0.06	0.00
PointGoal	0.01	**0.02**	0.01	0.01	0.02	0.00	0.00	0.00
	0.05	**0.06**	0.03	0.05	0.02	0.00	0.03	0.00
	0.10	**0.08**	0.04	0.04	0.02	0.00	0.03	0.00
	0.20	**0.23**	0.08	0.07	0.01	0.00	0.03	0.00
Swimmer	0.05	**0.02**	0.02	0.02	0.01	0.01	0.01	0.01
	0.10	**0.05**	0.03	0.03	0.01	0.01	0.01	0.01
	0.15	**0.28**	0.02	0.03	0.02	0.01	0.01	0.01

"Attack Power" denotes the perturbation magnitude κ. Attack efficiency is reported as the average episode cost per unit attack power.

collect 100 episodes (totalling 100,000 time steps) for mining the parameters of (10). The mined STL formula is

$$\phi_m = \mathbf{G}_{[0,1000]}\big((x > -1.008612) \wedge (x < 1.008612)\big), \tag{11}$$

whose parameters are very close to the ground-truth safety bounds $(-1, 1)$, demonstrating the accuracy of the mining procedure.

Safe PointGoal Navigation. In the PointGoal environment, the agent relies on LiDAR-type proximity sensors to perceive nearby obstacles. Each LiDAR channel $\mathrm{LiDAR}[i]$ encodes obstacle proximity (larger values indicate closer obstacles). We express a safety requirement via a pSTL formula that constrains the maximum allowable proximity:

$$\phi(d) = \mathbf{G}_{[0,T]}\left(\bigwedge_{i=1}^{n} \mathrm{LiDAR}[i] < d\right), \tag{12}$$

where $\mathrm{LiDAR}[i] < d$ enforces that all proximity readings remain below a threshold d, thereby ensuring a sufficient safety margin to obstacles. We collect 100 episodes (approximately 100,000 steps) and apply a template-based STL mining method to optimize d. The resulting STL formula is

$$\phi_m = \mathbf{G}_{[0,1000]}\left(\bigwedge_{i=1}^{n} \mathrm{LiDAR}[i] < 0.9218\right). \tag{13}$$

Although the mined formula differs in form from the hand-crafted specification, it accurately captures the observed safety behavior, illustrating the flexibility and generalization capability of STL-based mining.

Safe SwimmerVelocity. For Safe SwimmerVelocity, we mine an STL formula that jointly constrains forward velocity and lateral position. Collecting 100 trajectories from the environment, the mined specification is

$$\phi_3 = \mathbf{G}_{[0,1000]}\begin{pmatrix} v_x < 0.2265 \ \wedge \\ y_{\min} \geq -0.935836 \wedge \\ y_{\max} \leq 1.012948 \end{pmatrix}, \tag{14}$$

where $y_{\min}$ and $y_{\max}$ denote the minimum and maximum lateral positions observed within an episode, respectively. The mined parameters closely match the true safety bounds on velocity (0.2282 m/s) and position ($[-1, 1]$ m), indicating precise mining performance.

5.5 STLMAA Performance

Using the mined STL formulas, we deploy STLMAA against the trained safe RL agents under the four attacker settings: white-box, gray-box-π, gray-box-f, and black-box. We consider two evaluation protocols: (1) *safety termination*, where episodes terminate upon the first constraint violation and we report the violation rate; (2) *cost accumulation*, where episodes continue after violations and we report the average cumulative cost. Each attack configuration is evaluated over 200 episodes for perturbation magnitudes $\kappa \in \{0.01, 0.05, 0.10, 0.20\}$ (or up to 0.15 for Swimmer). Results are summarized in Fig. 3 and Table 2.

Across all environments, STLMAA consistently outperforms baseline attacks, with the following key observations:

(1) White-box STLMAA achieves the strongest attacks. In Safe PointCircle and Safe SwimmerVelocity, the white-box variant yields violation rates exceeding 0.9 and average costs above 400 and 100, respectively, at $\kappa = 0.20$ (PointCircle) and $\kappa = 0.15$ (Swimmer). In PointGoal, white-box attacks reduce the safety rate to 10% at $\kappa = 0.20$, with the average cost exceeding 60.

(2) Gray-box and black-box STLMAA scale effectively with attack strength. Although they are weaker than white-box attacks, both gray-box variants and the black-box attacker significantly outperform the random, gradient-based, and Max-cost baselines. Interestingly, the black-box attacker occasionally surpasses gray-box-f or gray-box-π, highlighting strong adaptability even with limited knowledge.

(3) STLMAA exhibits higher attack efficiency. As shown in Table 2, we define *attack efficiency* as the average episode cost per unit attack power (perturbation magnitude κ). STLMAA achieves substantially higher efficiency—for example, 0.23 in PointGoal and 0.28 in Swimmer—representing a 5×–10× improvement over the baselines. This indicates that STLMAA can discover critical vulnerabilities using relatively small perturbations per episode.

Overall, STLMAA demonstrates robust effectiveness and efficiency across different environments and attacker settings, confirming its practical impact in degrading safety and aligning with the theoretical robustness deviation bounds.

Experimental Verification of Theoretical Bounds. We further validate the theoretical robustness deviation bound in the Safe

PointCircle environment. Using estimated Lipschitz constants ($L = 0.6772$, $L_s = 5.3894$, $L_p = 1.0$, $K_\phi = 1.0$) and perturbation magnitude $\kappa = 0.03$, the predicted upper bound at $\delta = 0.1$ is 0.3941. Empirical evaluation over 1,000 perturbed samples shows that 89.1% of the deviations lie within this bound, with a mean deviation of 0.2085. The small fraction of samples that slightly exceed the bound can be attributed to numerical approximations in estimating Lipschitz constants and robustness values, but overall the results provide strong empirical support for the theoretical analysis.

6 Discussions

Limitations. First, compared to conventional adversarial attack approaches that directly perturb states or actions and often assume access to exact system gradients, STLMAA requires an initial trajectory collection phase through real or simulated interactions with the safe RL system. This data acquisition step is essential both for mining the parametric STL (pSTL) safety specification and for training the surrogate models used during attack generation. However, in practical deployments, continuous interaction may be expensive, time-consuming, or even infeasible—for example, in safety-critical CPS where exploratory rollouts risk violating operational constraints. This requirement may therefore limit the applicability of STLMAA in scenarios with restricted access to system trajectories.

Second, the pSTL template used to approximate the system's safety constraints is manually constructed based on domain knowledge and the distribution of collected trajectories. Such a handcrafted template may only provide an imperfect abstraction of the true underlying safety requirement. If the resulting mined formula ϕ_m deviates from the actual constraint boundary—either being overly conservative or insufficiently expressive—the attack may fail to expose the most critical vulnerabilities or may instead overestimate unsafe regions. Consequently, the fidelity of the pSTL representation becomes a key factor influencing both the effectiveness and the reliability of STLMAA.

A third limitation lies in the use of manually designed pSTL templates. In this work, we follow prior template-based STL mining methods and assume that a reasonable template structure can be specified from high-level prior knowledge of the task and key state variables. While this assumption is realistic in many engineering domains where safety requirements are documented or partially known, it also restricts the expressiveness of the mined specifications. An interesting direction for future work is to automatically suggest or refine template structures from data, for example by first clustering trajectories to identify candidate predicates and then searching over temporal operators, or by leveraging grammar-based or neuro-symbolic synthesis.

Potential Defense Methods. To strengthen the robustness of safe RL systems against STLMAA-style adversarial perturbations, several defense strategies may be considered. One promising direction is robust adversarial training, where perturbations—either synthetically generated or derived from STLMAA—are incorporated into the training loop. By exposing the agent to a diverse set of safety-critical perturbations during learning, the policy can acquire a more resilient understanding of the constraint boundary and better generalize to previously unseen disturbances.

Another defense mechanism is to design or reformulate system safety constraints in a way that makes them inherently more challenging to capture via STL mining. For instance, safety requirements that involve multi-modal conditions, hidden state dependencies, or non-Markovian properties may resist accurate reconstruction by existing STL mining algorithms. By embedding such structural complexity into the constraint design, defenders can increase the difficulty for attackers to infer precise safety boundaries, thereby reducing the vulnerability surface available to STL-guided attacks.

7 Conclusions

In this paper, we have shown that STL mining offers a rigorous framework for exploring vulnerabilities in safe RL-based systems. By extracting safety constraints through mined STL formulas and leveraging them to guide adversarial attacks, we have demonstrated that even advanced safe RL controllers remain susceptible to carefully crafted perturbations. Our theoretical analysis has established formal attack bounds, and our experiments across three environments have confirmed the efficiency and effectiveness of the proposed approach compared to existing methods. These findings have underscored the importance of integrating formal logic into the design of robust reinforcement learning systems.

Acknowledgment

This work was supported in part by NSF CNS-2333980 and CNS 2442914. 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] Erfan Aasi, Mingyu Cai, Cristian Ioan Vasile, and Calin Belta. 2023. Time-incremental learning of temporal logic classifiers using decision trees. In *Learning for Dynamics and Control Conference*. PMLR, 547–559.

[2] Edgar A Aguilar, Ezio Bartocci, Cristinel Mateis, Eleonora Nesterini, and Dejan Ničković. 2023. Mining specification parameters for multi-class classification. In *International Conference on Runtime Verification*. Springer, 86–105.

[3] Eugene Asarin, Alexandre Donzé, Oded Maler, and Dejan Nickovic. 2011. Parametric identification of temporal properties. In *International Conference on Runtime Verification*. Springer, 147–160.

[4] Ezio Bartocci, Cristinel Mateis, Eleonora Nesterini, and Dejan Ničković. 2023. Mining hyperproperties using temporal logics. *ACM Transactions on Embedded Computing Systems* 22, 5s (2023), 1–26.

[5] Giuseppe Bombara and Calin Belta. 2021. Offline and online learning of signal temporal logic formulae using decision trees. *ACM Transactions on Cyber-Physical Systems* 5, 3 (2021), 1–23.

[6] Yongchao Chen, Rujul Gandhi, Yang Zhang, and Chuchu Fan. 2023. Nl2tl: Transforming natural languages to temporal logics using large language models. *arXiv preprint arXiv:2305.07766* (2023).

[7] Tianyang Duan, Zongyuan Zhang, Zheng Lin, Yue Gao, Ling Xiong, Yong Cui, Hongbin Liang, Xianhao Chen, Heming Cui, and Dong Huang. 2025. Rethinking adversarial attacks in reinforcement learning from policy distribution perspective. *arXiv preprint arXiv:2501.03562* (2025).

[8] Georgios E Fainekos and George J Pappas. 2006. Robustness of temporal logic specifications. In *International Workshop on Formal Approaches to Software Testing*. Springer, 178–192.

[9] Jialiang Fan and Fanxin Kong. 2025. Survey of signal temporal logic specification mining: techniques, applications, and future directions. In *Disruptive Technologies in Information Sciences IX*, Vol. 13480. SPIE, 34–41.

[10] Tairan He, Chong Zhang, Wenli Xiao, Guanqi He, Changliu Liu, and Guanya Shi. 2024. Agile but safe: Learning collision-free high-speed legged locomotion. *arXiv preprint arXiv:2401.17583* (2024).

[11] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. 2017. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284* (2017).

[12] Susmit Jha, Ashish Tiwari, Sanjit A Seshia, Tuhin Sahai, and Natarajan Shankar. 2017. Telex: Passive stl learning using only positive examples. In *International Conference on Runtime Verification*. Springer, 208–224.

[13] 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. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. https://openreview.net/forum?id=WZmlxIuIGR

[14] Jiaming Ji, Jiayi Zhou, Borong Zhang, Juntao Dai, Xuehai Pan, Ruiyang Sun, Weidong Huang, Yiran Geng, Mickel Liu, and Yaodong Yang. 2024. Omnisafe: An infrastructure for accelerating safe reinforcement learning research. *Journal of Machine Learning Research* 25, 285 (2024), 1–6.

[15] Shixiong Jiang, Mengyu Liu, and Fanxin Kong. 2024. Backdoor attacks on safe reinforcement learning-enabled cyber–physical systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 43, 11 (2024), 4093–4104.

[16] Shixiong Jiang, Mengyu Liu, and Fanxin Kong. 2024. Vulnerability Analysis for Safe Reinforcement Learning in Cyber-Physical Systems. In *2024 ACM/IEEE 15th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 77–86.

[17] Karen Leung, Nikos Aréchiga, and Marco Pavone. 2023. Backpropagation through signal temporal logic specifications: Infusing logical structure into gradient-based methods. *The International Journal of Robotics Research* 42, 6 (2023), 356–370.

[18] Yongyuan Liang, Yanchao Sun, Ruijie Zheng, and Furong Huang. 2022. Efficient adversarial training without attacking: Worst-case-aware robust reinforcement learning. *Advances in Neural Information Processing Systems* 35 (2022), 22547–22561.

[19] Mengyu Liu, Pengyuan Lu, Xin Chen, Oleg Sokolsky, Insup Lee, and Fanxin Kong. 2024. Deadline-Safe Reach-Avoid Control Synthesis for Cyber-Physical Systems with Reinforcement Learning. In *2024 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 96–108.

[20] Mengyu Liu, Pengyuan Lu, Xin Chen, Oleg Sokolsky, Insup Lee, and Fanxin Kong. 2024. Model-free PAC Time-Optimal Control Synthesis with Reinforcement Learning. In *2024 22nd ACM-IEEE International Symposium on Formal Methods and Models for System Design (MEMOCODE)*. IEEE, 34–45.

[21] 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).

[22] 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, 152–166.

[23] Sara Mohammadinejad, Jyotirmoy V Deshmukh, and Aniruddh G Puranic. 2020. Mining environment assumptions for cyber-physical system models. In *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 87–97.

[24] Daniele Nicoletti, Samuele Germiniani, and Graziano Pravadelli. 2024. Mining signal temporal logic specifications for hybrid systems. In *2024 Forum on Specification & Design Languages (FDL)*. IEEE, 1–8.

[25] 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).

[26] Federico Pigozzi, Eric Medvet, and Laura Nenzi. 2021. Mining road traffic rules with signal temporal logic and grammar-based genetic programming. *Applied Sciences* 11, 22 (2021), 10573.

[27] Alex Ray, Joshua Achiam, and Dario Amodei. 2019. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708* 7, 1 (2019), 2.

[28] Maxwell Standen, Junae Kim, and Claudia Szabo. 2025. Adversarial Machine Learning Attacks and Defences in Multi-Agent Reinforcement Learning. *Comput. Surveys* 57, 5 (2025), 1–35.

[29] 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.

[30] 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*, Vol. 34. 5883–5891.

[31] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 5026–5033.

[32] Xiaoyan Wang, Yujuan Zhang, and Lan Huang. 2024. Improve Robustness of Safe Reinforcement Learning Against Adversarial Attacks. In *2024 4th International Conference on Electronic Information Engineering and Computer Science (EIECS)*. IEEE, 723–727.

[33] Lunet Yifru and Ali Baheri. 2024. Concurrent learning of control policy and unknown safety specifications in reinforcement learning. *IEEE Open Journal of Control Systems* (2024).

[34] 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* 33 (2020), 21024–21037.