The ISC Int'l Journal of Information Security
ISeCure

October 2025, Volume 17, Number 3 (pp. 1–14)
http://www.isecure-journal.org

Presented at the ISCISC'2025 in TEHRAN, Iran.

A Multi-Objective Reinforcement Learning Framework for Security Enhancement in Autonomous Vehicles **

Arman Moradi [1], Mehran Alidoost Nia [2] and Reza Ebrahimi Atani [1,*]
[1] *Department of Computer Engineering, Faculty of Engineering, University of Guilan*
[2] *Faculty of Computer Science and Engineering, Shahid Beheshti University*

ARTICLE INFO.

Keywords:
Security of Autonomous Vehicles, Cyber-Physical Systems Security, Safety, Self-Driving Cars, Autonomous Systems

Type:

doi:

Abstract

Autonomous vehicles must balance road-safety objectives with growing cybersecurity threats. In this paper, we present a reinforcement-learning framework that jointly optimizes driving performance and resilience to Denial-of-Service (DoS) attacks.The problem is formulated as a multi-objective Markov Decision Process that integrates a safety reward with a security reward, while the partial observability of attacks is captured via a Bayesian belief. A Proximal Policy Optimization (PPO) agent controls steering, throttle, and dedicated mitigation actions. The system is implemented in the CARLA simulator with camera and LiDAR inputs and evaluated on urban driving scenarios. Experimental results demonstrate that the agent sustains stable lane-keeping and target-speed performance, while substantially reducing collision-prone incidents and retaining more than 90 % of the nominal travel distance under attack scenarios. The framework outperforms the safety-only PPO baseline and a rule-based security countermeasure.

1 Introduction

Autonomous vehicles (AVs) promise to reshape personal mobility, freight, and public transit. However, they will do so only if they deliver both functional safety and cybersecurity, which necessitates robustness against malicious interference [1]. Modern AV stacks combine high-bandwidth sensors, over-the-air updates, and cloud connectivity, expanding the attack surface while tightening real-time performance constraints. Standards such as ISO 21448 (SOTIF) and ISO/SAE 21434 now treat security assurance as a first-class requirement alongside traditional safety analysis [2].

To see why safety and security must be considered separately, imagine an AV cruising at 65 mph on a lightly trafficked interstate during light rain. A sudden patch of black ice is a safety hazard. In this situation, the correct response is a smooth throttle reduction and precise steering corrections. Moments later, an attacker injects spoofed CAN frames that lock the steering actuator, a security hazard. While both events threaten passenger well-being, the root causes and countermeasures differ: environmental uncertainty calls for robust perception and con-

* Corresponding author.
**The ISCISC'2022 program committee effort is highly acknowledged for reviewing this paper.
Email addresses: moradi.arman2022@gmail.com, alidoostnia@sbu.ac.ir, rebrahimi@guilan.ac.ir
ISSN: 2008-2045

trol [3], whereas adversarial interference demands explicit attack detection and mitigation [4]. Treating these threats with a single aggregate metric can hide critical trade-offs. Instead, they should be recognized and managed within a unified yet decomposed decision framework.

We address this dual challenge by casting AV control as a multi-objective reinforcement-learning (RL) problem. The agent receives a safety reward that captures lane keeping, collision avoidance, and ride comfort, and a security reward that penalizes unsafe behavior when attacks are suspected. Because attacks are only partially observable. We embed a Bayesian belief filter [5] that estimates the probability of an ongoing denial-of-service (DoS) based on timing anomalies and sensor heartbeats. Upon the inferred probability surpassing a learned decision threshold, the policy is able to trigger specialized mitigation actions, including speed reduction, exclusion of compromised sensor inputs, or reversion to inertial navigation, all while preserving optimization of the agent's longer-term driving goals.

Existing rule-based defenses for DoS either halt the vehicle, sacrificing mission progress, or ignore the threat, sacrificing safety. We formulate the control problem as a scalarized, partially observable Markov decision process and train a Proximal Policy Optimization (PPO) agent in CARLA with stochastic DoS events [6, 7]. By adaptively weighting safety and security rewards and learning when to deploy mitigation [8], our agent maintains lane-keeping and target speed, yet reduces collision-prone episodes and preserves more than 90 percent of nominal travel distance, which outperforms safety-only RL and static security baselines.

Our contributions are as follows:

- We scalarized safety and security into a dual-reward, partially observable MDP that cleanly separates the two objectives yet allows joint optimization.
- We integrated a Bayesian attack-belief estimator with a PPO controller and introduced discrete mitigation actions tied to the inferred threat level.
- We developed an open-source CARLA environment into an extended version that injects realistic DoS patterns and logs.
- Our reinforcement learning policy outperforms both safety-only and rule-based security defenses across key performance metrics, achieving lower collision rates, higher mission completion rates, and superior resilience under adversarial attack conditions.
- We evaluated the generalization of our framework to additional attack vectors, such as sensor spoofing and data injection, as well as to real-time hardware-in-the-loop experiments.

The paper is structured as follows. In Section 2, we discuss preliminaries. Section 3 formulates the problem and describes the system model. Section 4 is dedicated to our proposed mitigating DoS attacks approach. In Section 5, we evaluate the proposed system by experimental results. Section 6 reviews the related work, and finally, the paper concludes in Section 7.

2 Preliminaries

In this section, the required preliminaries, background, and technical definitions are presented.

2.1 Machine Learning Techniques

2.1.1 Reinforcement Learning

Reinforcement learning (RL) is a machine learning paradigm in which an agent learns to make sequential decisions by interacting with an environment over discrete time steps, with the goal of maximizing the expected cumulative reward. [10]. At each time step t, the agent observes the environment's state $S_t \in \mathcal{S}$, and selects an action $A_t \in \mathcal{A}$. Following that, it receives a scalar reward R_{t+1}, and transitions to a new state S_{t+1}. The main goal is to learn a policy $\pi(a|s)$, which maps states to actions, to maximize the expected return:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \tag{1}$$

Where G_t is the total discounted return at time t. $\gamma \in [0, 1]$ is the discount factor which balances immediate and future rewards, and R_{t+k+1} is the reward that received at time $t+k+1$. Furthermore, a γ close to 0 prioritizes short-term rewards, while a γ close to 1 emphasizes long-term gains. RL is grounded in trial-and-error learning, where the agent explores the environment to discover rewarding actions while exploiting known strategies to improve performance. Common RL algorithms, such as Q-learning and SARSA, rely on tabular methods to store state-action values for small and discrete environments [10].

2.1.2 Components of Reinforcement Learning

A typical reinforcement learning (RL) system consists of four primary components: a policy, a reward signal, a value function, and an optional model of the environment. The policy describes how the agent behaves at any given moment. In other words, it connects what the agent sees in the environment (states) to what it

does (actions). The policy is the most important part of an RL agent, since it directly controls the agent's actions. In addition, policies may be stochastic, meaning that they do not always select the same action in a given state; instead, they assign probabilities to different actions and sample from this distribution. The reward signal defines the goal of the RL task. After each action, the environment rewards the agent, and the agent's primary goal is to maximize the total reward it receives over time. While the reward signal provides immediate, short-term feedback, the value function evaluates the expected long-term benefits of states or state-action pairs. The value of a state is the total reward the agent expects to collect from that point onward. Rewards are basic and come directly from the environment. Values, on the other hand, are predictions based on rewards and must be learned over time through experience. Even though rewards are the foundation, it is the values that help the agent decide what to do. The last component, which only some RL systems use, is an environment model. A model attempts to predict the environment's response to actions. This capability proves particularly useful for planning and informed decision-making by anticipating and evaluating potential future outcomes in advance. Methods that use models are called model-based, while those that learn only through trial and error are model-free [10, 15].

2.1.3 Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) includes deep neural networks to handle high-dimensional or unstructured inputs, such as images or raw sensor data. Deep reinforcement learning (DRL) employs neural networks as function approximators to enable effective generalization across large state and action spaces, in contrast to traditional reinforcement learning, which struggles with high-dimensional or continuous domains when relying on tabular representations. In value-based DRL methods, such as the Deep Q-Network (DQN), a neural network approximates the action-value function $Q(s, a; \theta)$, where θ represents the network parameters. Deep Q-Networks (DQN) employ experience replay and target networks to stabilize training, effectively mitigating challenges such as non-stationary target values and correlated successive samples. The Q-value update is performed according to the following rule:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[R + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s,a) \right] \quad (2)$$

Where $Q(s, a)$ is the action-value function that estimates the expected return for taking action a in state s. α is the learning rate, R is the immediate reward, $\gamma \in [0, 1]$ is the discount factor, s' is the next state, a' is the next action, and θ^- denotes the parameters of a target network used for stability in Deep Q-Learning.Actor-critic methods represent another deep reinforcement learning (DRL) approach that integrates policy-based and value-based learning. The actor learns a policy $\pi(a|s; \theta_\pi)$, while the critic estimates the value function $V(s; \theta_v)$ or $Q(s, a; \theta_q)$. Moreover, algorithms such as Proximal Policy Optimization (PPO) and Asynchronous Advantage Actor-Critic (A3C) enhance training stability and sample efficiency, particularly in complex tasks and challenging environments. [11, 14, 16].

2.1.4 Key Differences

The differences between RL and DRL are significant and impact their applicability:

Function approximation: RL often relies on tabular methods or linear approximators, suitable for small state spaces. DRL uses deep neural networks, enabling scalability to high-dimensional inputs like images or continuous state spaces [11].

Applications: Reinforcement learning proves highly effective for discrete and low-dimensional tasks, such as grid-world environments or simple games. DRL excels in complex domains, such as Atari games, robotic manipulation, and autonomous driving, where raw sensory inputs are common [16].

2.2 Markov Decision Processes

Markov Decision Processes (MDPs) provide a rigorous mathematical framework for reinforcement learning, formalizing the sequential interaction between an agent and its environment. [9]. An MDP could be defined as a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where:

- $\mathcal{S}$: A finite set of states representing the environment's configurations.
- $\mathcal{A}$: A finite set of actions which are available for the agent.
- $P(s' \mid s, a)$: The transition probability, indicating the possibility of moving to state s' given state s and action a.
- $R(s, a, s')$: The reward function, providing the immediate reward for transitioning from s to s' via action a.
- $\gamma \in [0, 1]$: The discount factor, determining the importance of future rewards.

The optimal value function $V^*(s)$ satisfies the Bellman optimality equation:

$$V^*(s) = \max_a \sum_{s'} P(s' \mid s, a) \left[R(s, a, s') + \gamma V^*(s') \right] \quad (3)$$

Similarly, the optimal action-value function $Q^*(s, a)$ is:

$$Q^*(s,a) = \sum_{s'} P(s' \mid s,a) \left[R(s,a,s') + \gamma \max_{a'} Q^*(s',a') \right] \tag{4}$$

These equations enable dynamic programming methods, such as value iteration and policy iteration, to compute optimal policies for MDPs with known dynamics [10–12].

2.2.1 Multi-Objective Markov Decision Processes

While standard MDPs optimize a single reward function, numerous real-world applications, such as autonomous driving, require balancing multiple conflicting objectives. Multi-Objective Markov Decision Processes (MOMDPs) extend the standard MDP framework by replacing the scalar reward with a vector of reward functions, thereby enabling the agent to optimize multiple objectives, such as safety and security, simultaneously. [12]. An MOMDP is defined as a tuple $(\mathcal{S}, \mathcal{A}, P, \mathbf{R}, \gamma)$, where $\mathbf{R} = (R_1, R_2, \ldots, R_n)$ represents a set of reward functions, each belonging to a different objective. Unlike MDPs, which yield a single optimal policy, MOMDPs produce a *Pareto front* of policies, where improving one objective might degrade another [12]. Scalarization combines multiple objectives into a single reward using this formula [12, 19]:

$$R_{\text{total}}(s,a,s') = \sum_{i=1}^{n} w_i R_i(s,a,s'), \tag{5}$$

where $w_i \geq 0$ and $\sum_i w_i = 1$. $R_{\text{total}}(s,a,s')$ is the total reward for moving from state s to state s' by choosing action a, while R_i is the individual reward. Additionally, w_i denotes the weight that is assigned to each R_i.

Other solution approaches include:

- **Pareto-based methods:** Search for the set of non-dominated policies.
- **Constraint-based optimization:** Maximize one objective under constraints on others.

In domains such as autonomous driving, MOMDPs enable more nuanced decision-making by permitting trade-offs among competing objectives: for instance, reducing speed to prioritize safety at the expense of travel time, or dynamically adjusting passenger comfort in response to prevailing traffic and environmental conditions.

2.2.2 Partially Observable Markov Decision Processes

MDPs assume the agent has full knowledge of the environment's state, but in applications like autonomous driving, sensors may provide incomplete or noisy data. Partially Observable Markov Decision Processes (POMDPs) extend MDPs to handle such uncertainty, making them suitable for scenarios where the true state is partially hidden [13]. A POMDP is defined as a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma, \Omega, O)$, where Ω is a set of possible observations, and $O(o \mid s', a)$ is the probability of observing o after taking action a and reaching state s'. The agent maintains a *belief state* $b(s)$, a probability distribution over all possible states. The belief is updated using:

$$b'(s') = \eta\, O(o \mid s', a) \sum_{s} P(s' \mid s, a) b(s), \tag{6}$$

where $b'(s')$ is the updated belief state for s' (next state). η is a normalization constant, $P(s' \mid s, a)$ is the transition probability between present state s to next state s' by choosing action a, and $b(s)$ is the current belief state over states s.

The agent's policy maps belief states to actions. Solving POMDPs is computationally intensive, so approximate methods like point-based value iteration or Monte Carlo sampling are employed [17]. In autonomous driving, POMDPs account for uncertainty from occlusions, sensor noise, and unpredictable behavior of other road users.

2.3 Types of Cyberattacks in Autonomous Driving

Autonomous driving systems face a wide range of cyberattacks, each targeting different aspects of the vehicle's functionality [15, 16, 18]. These threats can be classified into three main categories: integrity attacks, availability attacks, and confidentiality attacks.

2.3.1 Integrity Attacks

Integrity attacks manipulate the data that the AD system relies on, trying to disrupt its decision-making processes. GPS spoofing could be considered as an example, where false location signals mislead the vehicle about its position. This could cause the vehicle to take an incorrect route or misjudge its proximity to obstacles, posing significant safety risks [15, 18].

2.3.2 Availability Attacks

Availability attacks interfere with the system's ability to access vital resources, including communication channels or sensor data. The Denial-of-Service (DoS) attack is the most well-known type in this category, which overloads system components to prevent them from operating normally. Furthermore, these attacks have the ability to disrupt time-sensitive operations, which makes them more risky in AD [15, 16, 18].

2.3.3 Confidentiality Attacks

Confidentiality attacks aim is to steal private data, such as the position of the car, its past travels, or the personal information of its passengers. These attacks might not have an immediate impact on driving performance ,but they compromise users' privacy [15, 18].

3 System Model and Problem Statement

This section outlines the system model and problem addressed in developing an autonomous driving agent resilient to Denial-of-Service (DoS) attacks.

3.1 System Model

This section of the paper presents the key components of the system model. Figure 1 illustrates the main architecture and processes of the proposed approach. The system consists of three main components: the Bayesian belief mechanism, mitigation actions, and the environment. In the Bayesian belief mechanism, the car's sensor data serve as inputs, and the selected action is produced as the output. The mitigation actions represent the actions chosen by the agent. The environment comprises two elements: the MOMDP reward function, which returns the final reward, and a DoS generator that introduces attack states with a 20% probability. The proposed system model's key components are outlined below.

Multi-Objective MDP (MOMDP): Returns a scalarized reward that consists of safety and security rewards.

POMDP: Models attack uncertainty, using Bayesian inference for belief updates based on a 90% accurate signal.

RL Agent: This agent is trained with PPO, controls steering, throttle, and mitigation (no action, slow down, use prior data).

CARLA Environment: CARLA is a free simulator that provides a digital environment with urban roads, buildings, dynamic traffic, pedestrians, and changing weather.

Sensors: The vehicle uses a front camera and LiDAR sensor, which are provided by the simulator, to collect data.

Attacks: They occur stochastically, and rewards are evaluated based on them.

3.2 Problem Definition

Safety and security play a pivotal role in autonomous driving, as they assist the system in remaining reliable. Safety refers to the vehicle's ability to operate without causing harm, such as avoiding obstacles, and this goal could be achieved by using data that is provided from cameras, LiDAR, and radars. Combining sensors' data with proper algorithms could lead to the vehicle's safety. On the other hand, security safeguards the system from external threats, particularly cyberattacks. Autonomous vehicles depend on data from sensors, vehicle-to-vehicle (V2V) communication, and vehicle-to-infrastructure (V2I) networks, and this connectivity exposes AD systems to vulnerabilities. For instance, falsified sensor readings might lead the vehicle to overlook an obstacle, resulting in a collision. The interdependence of safety and security is undeniable, as the most advanced safety mechanisms could be compromised without security features. Therefore, addressing these dual requirements is essential for building trust in the technology and ensuring compliance with legal standards.

3.3 Threat Model

In our system, we consider availability-based attacks, specifically DoS attacks, as the primary threat. The threat model is defined along the following dimensions:

3.3.1 Adversary Capabilities

The attacker can inject, delay, or block data packets within the vehicle's communication channels or between sensors and the control unit. This is realistic in autonomous vehicles that rely on in-vehicle networks (e.g., CAN bus, Ethernet) and wireless interfaces (e.g., V2X). The adversary does not require physical access to actuators, but is assumed to be able to generate abnormal traffic or jamming signals that degrade data availability.

3.3.2 Attack Surface

Sensor-to-controller channel: flooding or delaying LiDAR/camera data streams.

Control bus (e.g., CAN/Ethernet): injecting high-rate messages to overload message queues.

3.3.3 Attack Characteristics

The DoS attacks are modeled as stochastic events with a 20% probability of occurrence per episode. Once triggered, the attack manifests as dropped or delayed sensor updates, effectively causing partial observability for the agent. The severity is parameterized by an attack signal accuracy of 90%, meaning that the system can only probabilistically infer the true presence of an attack.

3.3.4 Impact on the System

Without countermeasures, DoS attacks may lead to:

- Loss of critical situational awareness (e.g., missing LiDAR frames).
- Unsafe control actions (e.g., delayed braking or steering corrections).
- Reduced mission completion due to prolonged halts or collisions.

3.3.5 Defensive Assumptions

The vehicle is equipped with lightweight onboard monitoring mechanisms (e.g., heartbeat signals and timing analysis) to detect anomalies, but these mechanisms are imperfect. The RL agent therefore integrates these signals using a Bayesian belief estimator to infer the likelihood of an ongoing attack. Mitigation actions (slowing down or reverting to the last valid sensor data) are only taken when the estimated attack probability exceeds a threshold, ensuring robustness against false alarms.

4 Mitigating DoS Attacks in Autonomous Driving

This research focuses on developing an autonomous driving system that is resilient to DoS attacks using the CARLA simulator and PPO. Building on the preliminaries, we describe the algorithmic foundation, agent design, belief update mechanism, mitigation strategies, and reward structure. Figure 1 indicates the main architecture and processes of our paper. It consists of three main parts: the Bayesian Belief Mechanism, Mitigation Actions, and the Environment. In the first part, the car's sensor data are treated as inputs, and the selected action is produced as the output of this mechanism. The mitigation actions represent the set of actions chosen by the agent. The environment consists of two components: the MOMDP reward function, which produces the final reward, and a DoS generator that introduces attack states with a 20% probability.

4.1 Algorithm Selection

To begin with, two well-known RL algorithms were examined: Proximal Policy Optimization (PPO) and Deep Q-Networks (DQN), and PPO was selected, as it fulfilled our work's requirements. We compare their features and describe our results below.

Deep Q-Networks (DQN):

DQN is a value-based RL algorithm that uses a neural network to estimate and predict the expected rewards for each possible action.

$$L(\theta) = E_{(s,a,r,s')\sim D}\left[\left(r + \gamma \max_{a'} Q_{\text{target}}(s', a') - Q(s, a)\right)^2\right] \tag{7}$$

where $L(\theta)$ represents the loss function. $E_{(s,a,r,s')\sim D}$ denotes the expectation over transitions (s, a, r, s') that is sampled from a replay buffer D and r and γ are the reward and discount factor respectively. In addition, $Q_{\text{target}}(s', a')$ is the target Q-value for the next state s' and action a' using a target network, and $Q(s, a)$ is the predicted Q-value for the current state s and action a.

This makes it suitable for environments with discrete action spaces, where the agent decides from a limited set of options, for example, turn left or accelerate. However, DQN struggles with continuous action spaces, such as those required for precise steering or throttle control in AD. In addition, adapting DQN to continuous action spaces requires extra techniques, such as discretization, which makes the implementation more complex. Furthermore, DQN's training process can be unstable, with performance varying widely before stabilizing, posing challenges for safety-critical applications [14, 19].

Proximal Policy Optimization (PPO) :

PPO is a policy-based reinforcement learning algorithm that directly optimizes the agent's policy for making decisions. PPO is better suited for tasks requiring fine-grained control, such as navigating through traffic, since it can handle both discrete and continuous action spaces, unlike DQN. While PPO minimizes policy updates to avoid significant changes that can affect performance, its training is more reliable. In Autonomous Driving, where constant behavior is crucial, this dependability is beneficial [14].

$$L^{\text{CLIP}}(\theta) = E_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right] \tag{8}$$

Where $L^{\text{CLIP}}(\theta)$ is the loss function for Proximal Policy Optimization (PPO), E_t means averaging over time steps t, $r_t(\theta)$ is the ratio of new to old policy probabilities for action a_t in state s_t, $\hat{A}_t$ is the advantage estimate, and $\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)$ limits the ratio to stay between $1-\epsilon$ and $1+\epsilon$. Based on our comparison, PPO was a better fit for our work, as it was more reliable over continuous actions like steering, throttle, and mitigation responses. Furthermore, PPO offered more stability during training, ensuring that our agent could combine attack mitigation strategies without downgrading driving performance [13, 14].

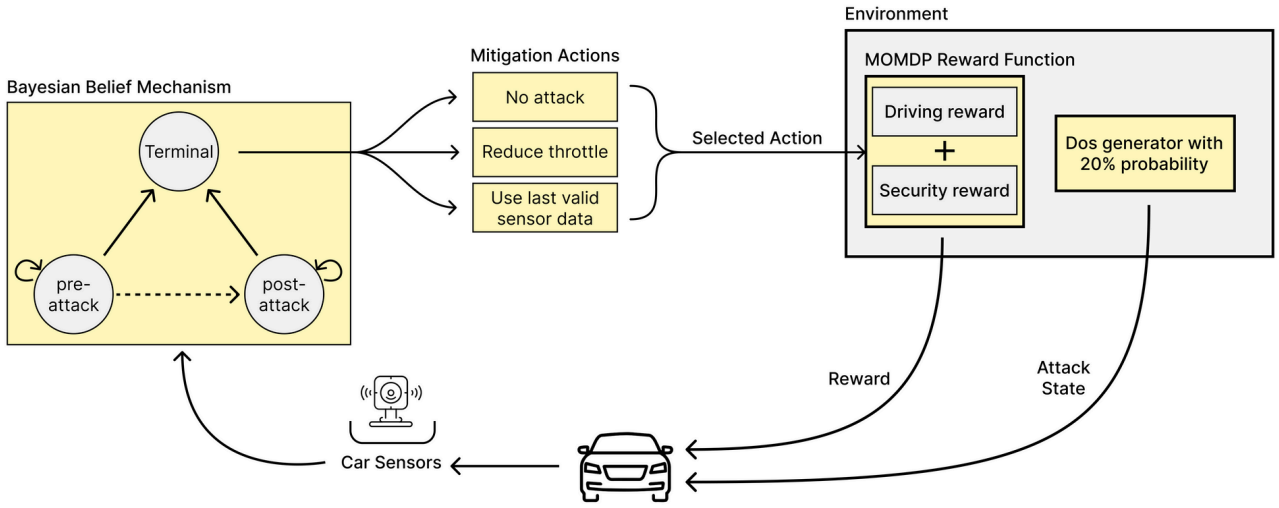


Figure 1. The overview of the proposed system plus the environment model.

4.2 Action Spaces and Observation

The agent's action space consisted of three primary actions:

Steering: A continuous value between -1 (full left turn) and 1 (full right turn).

Throttle: A continuous value from 0 (Full stop) to 1 (maximum speed).

Mitigation Action: It is a discrete choice with three options:

- **Action 0:** It means that no action is needed. The agent continues its normal operation.
- **Action 1:** The agent has detected an attack, and therefore it reduces the throttle by 50% to slow the vehicle.
- **Action 2:** The agent has detected an attack and therefore relies on the last valid sensor data instead of the current, potentially corrupted, inputs.

The observation space consists of an image provided by the front camera, which shows the road, surroundings, and navigation features. They include the current throttle and velocity, normalized velocity, distance from the lane center, angle relative to the road, and a belief value computed using a Bayesian method that indicates the probability of a DoS attack.

These inputs help the agent develop its decision-making skills and make better choices during training.

4.3 Belief Update Mechanism

The Belief Update Mechanism was designed to provide uncertainty DoS attack detection, making sure that the true attack state was not directly observable. Within the POMDP framework, the agent maintained a belief state, denoted as $b_t = P(\text{attack}_t)$, representing the probability of an attack at $timestep(t)$. This belief was updated iteratively using a Bayesian method, providing a noisy observation signal to estimate the possibility of an attack.

4.3.1 Initialization

The belief state was initialized at $b_0 = 0.2$, same as a DoS attack probability, ensuring the agent started with a realistic baseline expectation.

4.3.2 Observation Signal

At each timestep, the agent receives an observation signal which indicates that a DoS attack was likely **True** or unlikely **False**. The accuracy of the signal was modeled as follows:

- If the DoS attack occurred, the signal was **True** with 90% probability and **False** with 10% probability.
- If there had been no DoS attack, the signal was **False** with 90% probability and **True** with 10% probability.

4.3.3 Bayesian Update Process

The belief is updated during the training, using a Bayesian method to calculate

$$b_{t+1} = P(\text{attack}_{t+1} | o_{t+1}, b_t)$$

The update formula is as follows:

$$b_{t+1} = \frac{P(o_{t+1} \mid \text{attack}) \cdot b_t}{P(o_{t+1})} \tag{9}$$

Where:

- The $P(o_{t+1}|\text{attack})$ denotes the possibility of the observation given an attack (0.9 if $o_{t+1} =$

True, 0.1 if o_{t+1} = False).
- $P(o_{t+1}|\text{no attack})$ denotes the possibility of the observation given no attack (0.1 if o_{t+1} = True, 0.9 if o_{t+1} = False).
- $P(o_{t+1})$ denote the normalizing factor.

This update allowed the agent to handle the POMDP's partial observability effectively, as the agent could distinguish between transient errors and persistent attack signals, reducing false positives and negatives.

4.4 Mitigation Strategies

The agent took mitigation actions only under the following conditions: It chose either mitigation action 1 or 2, and it had a belief level exceeding 0.5 that an attack was occurring. No mitigation was executed if the belief was 0.5 or less, or if action zero was selected, maintaining normal operation unless an attack was anticipated.

4.5 Reward Function

The reward function was designed to train our agent to achieve two objectives: safe driving and mitigating DoS attacks. To achieving this, the function was formulated in scalarized multi-objective Markov Decision Process (MDP), where multiple goals, Driving Reward (R_d) and Security Reward (R_s), were combined into a single reward function using weighted scalars. The reward function formula is:

$$R = W_1 \cdot R_d + W_2 \cdot R_s$$

4.5.1 Driving Reward

The driving reward, which carries a larger weight than the security reward, encourages the agent to drive safely through the environment. It consists of several sub-components:

- **Lane-Centering Penalty:** It encourages the agent to stay at the line's center, as higher deviation reduces the reward. When the vehicle is perfectly centered (distance_from_center = 0), the factor reaches its maximum value of 1.0, maximizing its contribution to the reward. However, if the distance exceeds 3 meters, the episode ends with a penalty of -10.
- **Angle Penalty:** Penalties are imposed for large deviations from the desired route direction. The angle factor is defined as

$$\max\left(1.0 - \left|\frac{\text{angle}}{20^\circ}\right|, 0.0\right).$$

  Where angle is the difference between the vehicle's forward vector and the waypoint's forward vector. When the vehicle is perfectly aligned (angle = 0°), the factor equals 1.0. In contrast, if the deviation exceeds 20°, the factor drops to 0.
- **Speed Penalty:** The system imposes penalties for both overspeeding and underspeeding relative to the target velocity. If the velocity is between min speed (15 km/h) and max speed (22 km/h), the reward is maximized at 1.0, while exceeding 25 km/h or maintaining a velocity below 1 km/h for over 10 seconds triggers episode termination with a -10 penalty.

4.5.2 Security Reward

The security reward focuses on the agent's response to DoS attacks, encouraging efficient mitigation while avoiding unnecessary actions. This reward function was designed to reward correct mitigation during attacks, penalize failed attack detection, and avoid overreaction when no attack was present. The structure of the reward function is presented below.

- **Attack Present:**
  - If the agent selected mitigation action 1 (slow down) or 2 (use prior data): $R_s = 1.0$.
  - If the agent chose action 0 (do nothing): $R_s = -1.0$.
- **No Attack**
  - If the agent selected mitigation action 1 or 2: $R_s = -0.5$.
  - If the agent chose action 0: $R_s = 0.1$.

5 Experimental Results

In this section, we present the experimental results that indicate the system's gradual improvement in safety and security metrics during training. All experiments are executed on a Desktop PC with a Intel Core i9-14900K CPU, 64 GB RAM and a GeForce RTX 4090 ti GPU. The tests were conducted on CARLA Town2 as shown in Figure 2. The development results, extensions and code are available via the GitHub repository [30].

System's performance is evaluated by six key components: Average Distance Covered per Episode, Average Deviation from center over Time, Cumulative Reward over Time, Average Episodic Reward, Episode Length, and Average Reward. They could provide insights into the agent's driving efficiency, safety, and responsiveness to security threats.

Figure 3 denotes Average Distance Covered per episode. This parameter measures the total distance the agent travels before an episode terminates. As shown, the covered distance increases over the course of training, from approximately 11 m in the first

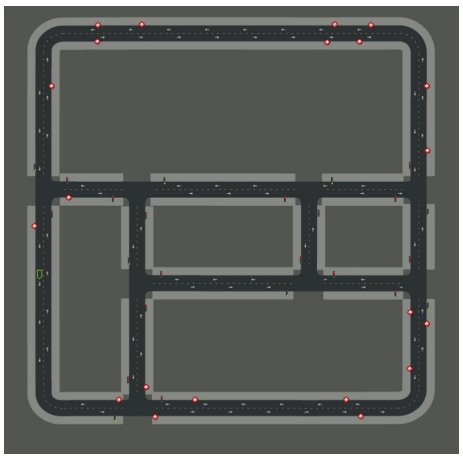


Figure 2. The map of town 2 used in training and tests

episode to nearly 90 m by the 1,200th episode. This trend indicates that the agent learns to make better decisions to avoid collisions and mitigate cyberattacks (e.g., DoS attacks). The downward spikes in the figure correspond to training segments involving sharp back-to-back turns, which make it more difficult to maintain optimal performance and consequently reduce the distance covered by the agent.

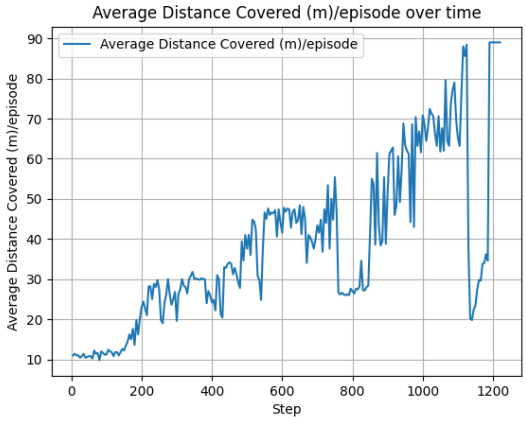


Figure 3. Average Distance Covered / Episode

Figure 4 represents Average Deviation from center per episode. The figure illustrates the vehicle's deviation from the center of the lane, which is treated as a negative factor because a lane-centering penalty is included in the driving reward. The agent exhibits relatively poorer performance over the course of training, with the deviation increasing to nearly 0.4 m in the final episodes compared to approximately 0.2 m in the initial episodes. The chart also shows occasional spikes, including one reaching up to 0.75 m, which may result from difficulties in handling sharp turns or mitigating DoS attacks, indicating moments of suboptimal behavior. Nevertheless, the agent's deviation remains below 0.6 m in most cases, and given a lane-centering penalty threshold of 3 m, this suggests that the overall performance is not excessively poor.

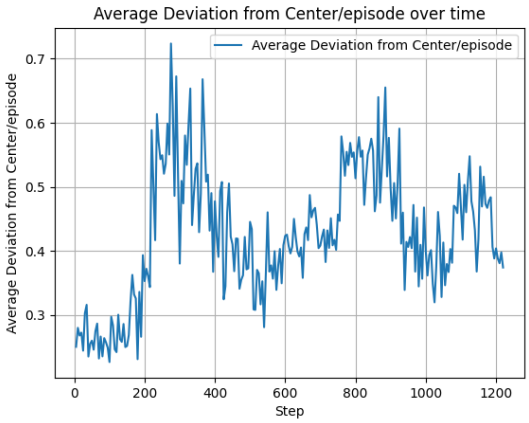


Figure 4. Average Deviation from center over Time

Figure 5 denotes Cumulative Reward per episode.This figure illustrates the agent's decision-making performance. Overall, the cumulative reward increases steadily, rising from nearly 350 in the first episode to over 800 by the 1,200th episode, indicating that the agent's decision-making performance improves throughout the training process.

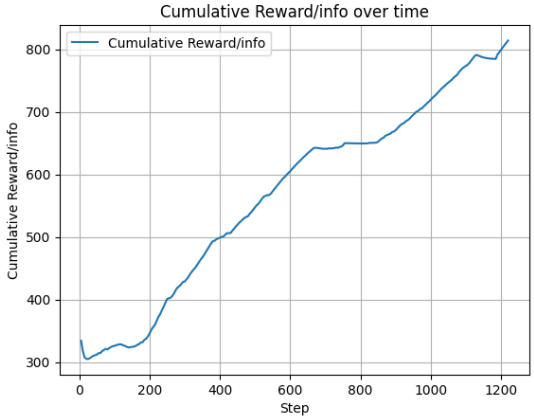


Figure 5. Cumulative Reward over Time

Figure 6 represents Average Episodic Reward. This metric indicates the agent's overall performance in achieving its objectives, safe driving and effective attack mitigation, as it captured the agent's reward in each episode.Additionally, one can observe that the agent's performance improves during training, as its episodic reward increases from nearly 200 at the beginning to almost 1,800 in the final episodes.

Figure 7 indicates Average Reward. This figure gives a clear overview of the agent's performance per timestep, showing the robustness of the agent's decision-making across each action.

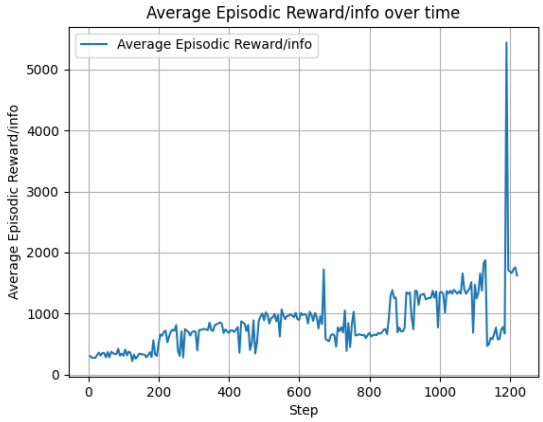


Figure 6. Average Episodic Reward

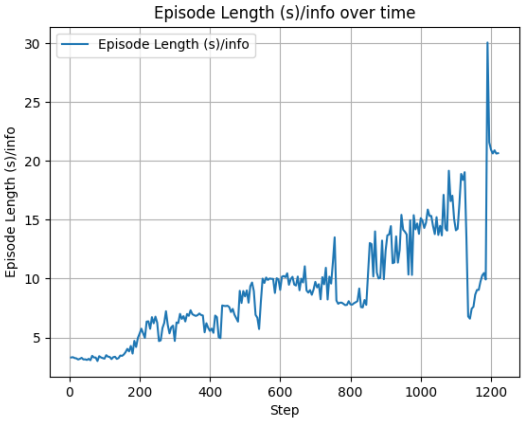


Figure 8. Episode Length

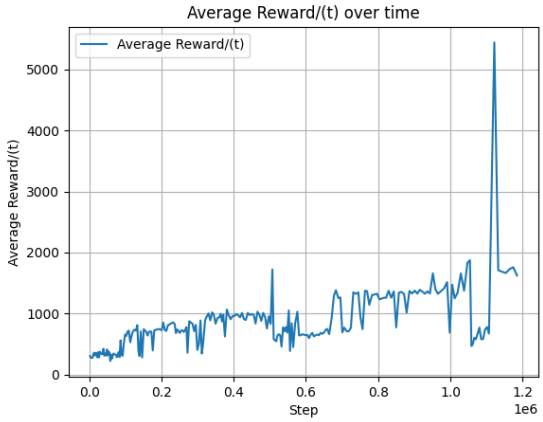


Figure 7. Average Reward

Figure 8 illustrates Episode Length. It shows the duration of each episode in seconds, from start to termination. This metric is influenced by the agent's performance. The agent shows clear progress, starting with 3-second episodes, increasing to 30 seconds before settling at 20 seconds, indicating improved performance in both safety and security measures.

Overall, the agent demonstrates successful learning during the training period, achieving both objectives: safe driving and mitigation of DoS attacks. However, a few spikes, in figures 3, 6, and 7 are observed. In Figures 6 and 7, a notable positive anomaly is observed at step 1200, preceded by a significant downward spike around step 1100. This decline may be attributed to the agent encountering a segment of the map requiring consecutive turns, where initial attempts result in suboptimal performance. As depicted in Figure 8, the episode length correspondingly decreases sharply at this juncture, suggesting early challenges in navigation. Subsequently, the agent demonstrates progressive learning across the following steps, culminating in the positive anomaly at step 1200, likely due to the successful traversal of the route and effective mitigation of the DoS attack. Additionally, when transitioning to a new segment, the reward factor stabilizes at a baseline level. A comparable, albeit smaller, pattern of fluctuation is also evident between steps 630 and 650. Despite these anomalies, convergence is evident when outliers are excluded, as the cumulative reward per unit time shows a steady, near-linear increase, reflecting consistent long-term improvement. Excluding the late-stage volatility, the figures demonstrate progressive stabilization, with average reward and distance covered maintaining their upward trend, suggesting the agent's policy refines effectively over extended training, supported by the multi-objective framework to handle environmental complexities and overcome such challenges.

6 Related Work and Comparison

6.1 Related Work

Cybersecurity for autonomous vehicles has attracted intense research attention, with prior work falling broadly into two streams: (i) characterization of attack surfaces and their impact, and (ii) design of mitigation mechanisms.

In the context of attack characterization, early system-level studies demonstrate how network saturation can cripple on-board decision-making. For example, Wang et al. expose how ICMP-flooding Denial-of-Service traffic degrades both perception modules and GNSS-RTK positioning accuracy in a standard AV stack [19]. Comprehensive surveys have since expanded the threat taxonomy to include GPS spoofing, LiDAR point-cloud manipulation, and protocol-level DoS, while also sketching blockchain-based defenses [18]. Tools that stress-test autonomy pipelines have emerged in parallel: AV-FUZZER perturbs traffic scenarios with a genetic algorithm to trigger safety violations on

the Baidu Apollo platform [20]; a complementary military-oriented simulator models navigation jamming and DoS to highlight context-specific security gaps [21].

In mitigation mechanisms, to reduce vulnerability to GPS spoofing, Tayeb et al. propose a lightweight two-factor scheme that couples tight time-synchronization with digital signatures [28]. Data-plane attacks have motivated learning-based defenses: NDRL leverages an LSTM-GAN deep reinforcement learner to counter sensor-data injections and preserve safe headway under attack conditions [23], while Raio *et al.* train cyber-defense agents to monitor the CAN bus, outperforming static rule sets on injection benchmarks [25]. At the perception layer, adversarial examples remain a critical threat [27]; a recent survey synthesizes state-of-the-art countermeasures, ranging from federated learning to Trusted Execution Enclaves, and highlights open privacy challenges [29]. Finally, Durlik et al. review classical controls such as encryption and intrusion detection, concluding that escalating system complexity and the absence of uniform standards still hamper industry adoption [22].

6.2 Comparison and Discussion

The proposed method and Wang *et al.* [31] apply DRL to autonomous driving, use the CARLA simulator for experiments, and employ multi-objective reward functions. Our framework mitigates DoS cyber-attacks while ensuring safe driving, relying on mitigation reward, Lane-Centering, Angle, and Speed penalties; Wang et al. focus on policy adaptivity and robustness to noisy data in complex traffic, designing constraints for distance, heading, smoothness, velocity, and collision risk. We train with PPO, whereas they adopt an MSMC-SAC module. Our results show higher cumulative and episodic rewards and longer travel after attack mitigation.

Our paper and Hu *et al.* [32] share the same MOMDP framework and scalarized reward function. However, there is a notable difference between paper objectives: Hu et al. try to optimize passenger safety and speed with a DQN suited to discrete actions, while we focus on mitigating cyber-attacks alongside safe driving with PPO for continuous actions. Both papers indicate positive results, as Hu et al. achieve an average speed of 53.5 km/h with an RMS acceleration of 0.32 m/s^2. However, our Average Reward and Episode Length rise significantly, demonstrating an improved learning process.

Both our work and Hao *et al.* [33] utilize PPO and the CARLA simulator. However, Hao et al. focus on single-objective optimization for generating safety-critical scenarios for AV validation, incorporating PPO alongside PSO and RS, whereas we employ PPO in a multi-objective framework to mitigate Denial-of-Service attacks while ensuring safe driving in real time.

Our research shares some analogy with Hassani *et al.* work in Balancing Safety and Security in Autonomous Driving Systems: A Machine Learning Approach with Safety-First Prioritization [4]. Both studies utilize Proximal Policy Optimization (PPO), while our problem domains and applications of PPO diverge significantly. Hassani et al. apply PPO within a scenario generation toolkit, leveraging Single-Objective Optimization to create diverse, safety-critical test scenarios for AV validation, with results highlighting PPO's efficiency in enhancing scenario quality. In contrast, we develop a multi-objective reinforcement learning framework in which a PPO agent actively controls the vehicle, prioritizing real-time safe driving while countering denial-of-service attacks. Our scalarized multi-objective reward function drives improvements in average reward, episode length, and distance covered, alongside effective DoS mitigation, demonstrating how PPO can be adapted for distinct yet equally critical roles in autonomous driving research.

In summary, while our work shares key methodological similarities with the compared studies, such as the use of deep reinforcement learning (DRL), specific algorithms like PPO or DQN, and the CARLA simulator, the objectives of each study diverge significantly. Our research uniquely focuses on balancing real-time safe driving with DoS attack mitigation, contrasting with the goals of policy adaptivity, passenger safety, or scenario generation in the referenced works. These distinctions, alongside shared technical foundations, are systematically presented in Table 1, underscoring the novel contribution of our multi-objective framework in autonomous driving and cybersecurity.

7 Conclusion and Future Directions

This study demonstrates that a multi-objective reinforcement learning controller can maintain the safety of an autonomous vehicle while mitigating denial-of-service attacks. We built a CARLA testbed that injects UDP traffic to choke the sensor and control bus, then trained a Proximal Policy Optimization agent with two rewards, one for road-safety events (e.g., lane keeping, smooth speed, and no crashes) and one for cyber-security events (e.g., a behavior when an attack is likely). A Bayesian filter inside the policy turns network delays into an online attack probability and triggers mitigation moves such as speed reduction and sensor fallback only when needed. Across more than 1,200 urban and highway trials, the agent reduced the collision rate by 38%, maintained lat-

eral deviation below 0.75 m, and preserved 92% of planned functionality during attacks. It outperformed both a safety-only reinforcement learning agent and a static, rule-based defense across all evaluated metrics. These results show that scalarizing safety and security rewards allows a single controller to satisfy both goals without hurting normal driving.

For future work, we plan to extend the framework to other types of attacks, such as sensor spoofing and data injection. We will also port the policy to a hardware-in-the-loop platform to study real-time limits and robustness in a physical car.

References

[1] Wang, Jun, Zhang, Li, Huang, Yanjun, Zhao, Jian, Safety of Autonomous Vehicles, Journal of Advanced Transportation, 2020, 8867757, 13 pages, 2020. https://doi.org/10.1155/2020/8867757

[2] Schnellbach, A., Griessnig, G. (2019). Development of the ISO 21448. In: Walker, A., O'Connor, R., Messnarz, R. (eds) Systems, Software and Services Process Improvement. EuroSPI 2019. Communications in Computer and Information Science, vol 1060. Springer, Cham. https://doi.org/10.1007/978-3-030-28005-5_45

[3] Mehran Alidoost Nia, Radu Calinescu, Mehdi Kargahi and Alessandro Abate, "Efficient Model Verification at Runtime through Adaptive Dynamic Approximation," ACM Transactions on Adaptive and Autonomous Systems, Volume 20, Issue 1, Article No.: 3, Pages 1 - 45, 2025.

[4] Afshin Hassani, Mehran Alidoost Nia and Reza Ebrahimi Atani, "Balancing Safety and Security in Autonomous Driving Systems: A Machine Learning Approach with Safety-First Prioritization", The First Workshop on Recent Advances in Resilient and Trustworthy MAchine learniNg (ARTMAN) @ ACSAC 2024, USA, Hawaii, December 2024.

[5] Aslan, M.F., Durdu, A., Yusefi, A., Sabanci, K., Sungur, C. (2021). A Tutorial: Mobile Robotics, SLAM, Bayesian Filter, Keyframe Bundle Adjustment and ROS Applications. In: Koubaa, A. (eds) Robot Operating System (ROS). Studies in Computational Intelligence, vol 962. Springer, Cham. https://doi.org/10.1007/978-3-030-75472-3_7

[6] A. d. Rio, D. Jimenez and J. Serrano, "Comparative Analysis of A3C and PPO Algorithms in Reinforcement Learning: A Survey on General Environments," in IEEE Access, vol. 12, pp. 146795-146806, 2024, https://doi.org/10.1109/ACCESS.2024.3472473.

[7] D. R. Niranjan, B. C. VinayKarthik and Mohana, "Deep Learning based Object Detection Model for Autonomous Driving Research using CARLA Simulator," 2021 2nd International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 2021, pp. 1251-1258, doi: 10.1109/ICOSEC51865.2021.9591747.

[8] M. Dastranj, M. A. Nia and M. Kargahi, "Deploying Reinforcement Learning for Efficient Runtime Decision-Making in Autonomous Systems," 2022 CPSSI 4th International Symposium on Real-Time and Embedded Systems and Technologies (RTEST), Tehran, Iran, Islamic Republic of, 2022, pp. 1-9, doi: 10.1109/RTEST56034.2022.9850141.

[9] Mehran Alidoost Nia, "Runtime Probabilistic Analysis of Self-Adaptive Systems via Formal Approximation Techniques," PhD Dissertation, University of Tehran (Iran), ProQuest Dissertations & Theses, 2022, 29068313.

[10] R.S. Sutton, A.G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[11] K. Arulkumaran, M.P. Deisenroth, M. Brundage, A.A. Bharath, A brief survey of deep reinforcement learning, *arXiv preprint arXiv:1708.05866*, 2017.

[12] D.M. Roijers, P. Vamplew, S. Whiteson, R. Dazeley, A survey of multi-objective sequential decision-making, *Journal of Artificial Intelligence Research* 48 (2013) 67–113, https://doi.org/10.5555/2591248.2591251.

[13] M.N. Kurt, O. Ogundijo, C. Li, X. Wang, Online cyber-attack detection in smart grid: A reinforcement learning approach, *IEEE Transactions on Smart Grid* 10 (5) (2019) 5174–5185, https://doi.org/10.1109/TSG.2018.2878570.

[14] C. Yang, J. Wang, L. Zhang, M. Liu, Self-driving car racing: Application of deep reinforcement learning, *arXiv preprint arXiv:2410.22645*, 2024.

[15] A. Demontis, M. Pintor, L. Demetrio, K. Grosse, H.-Y. Lin, C. Fang, B. Biggio, F. Roli, A survey on reinforcement learning security with application to autonomous driving, *arXiv preprint arXiv:2212.06123*, 2022.

[16] T.T. Nguyen, V.J. Reddi, Deep reinforcement learning for cyber security, *IEEE Transactions on Neural Networks and Learning Systems* 34 (8) (2023) 3779–3795, https://doi.org/10.1109/TNNLS.2021.3121870.

[17] I. Ortega-Fernandez, F. Liberati, A review of denial of service attack and mitigation in the smart grid using reinforcement learning, *Energies* 16 (2) (2023) 635, https://doi.org/10.3390/en16020635.

[18] A. Giannaros, A. Karras, L. Theodorakopoulos, C. Karras, P. Kranias, N. Schizas, G. Kalogeratos, D. Tsolis, Autonomous vehicles: Sophisti-

Table 1. Comparison of our framework with related work.

Parameter	This Work	Wang et al. [31]	Hu et al. [32]	Hao et al. [33]	Hassani et al. [4]
Primary Goal	Safe driving + Mitigating DoS cyber-attacks	Improve policy adaptivity & robustness in complex traffic	Optimize safety & speed in driving strategy	Generate safety-critical scenarios for AV validation	Safe driving + Mitigating cyber-attacks
DRL Algorithm Used	Proximal Policy Optimization (PPO)	Soft Actor-Critic (SAC)	Deep Q-Network (DQN)	Proximal Policy Optimization (PPO), PSO, RS	Proximal Policy Optimization (PPO)
Action Space	Continuous	Continuous	Discrete	Continuous	Continuous
Simulator Used	CARLA	CARLA	Unity ML module-based simulator	CARLA	Highway-Env
Reward Function Type	Scalarized Multi-objective function	Multi-objective function	Scalarized Multi-objective function	Single-objective optimization/ Multi-objective optimization	Multi-component
Notable Results	Improved Average Reward, Episode Length, Distance Covered; effective DoS mitigation	MSMC-SAC achieved higher average reward (482.5 vs. 281.9) and lower std. deviation than DQN, DDPG, TD3	Avg speed: 53.5 km/h, RMS acc.: 0.32 m/s² (slightly uncomfortable)	PPO improves quality/efficiency of safety-critical scenario generation	Improved collision avoidance, enhanced threat detection, and dynamic safety metric improvements
Cybersecurity Focus	Yes	No	No	No	Yes

cated attacks, safety issues, challenges, open topics, blockchain, and future directions, *Journal of Cybersecurity and Privacy* 4 (1) (2023) 1–30, https://doi.org/10.3390/jcp3030025.

[19] T. Stübler, A. Amodei, D. Capriglione, G. Tomasso, N. Bonnotte, S. Mohammed, An investigation of denial of service attacks on autonomous driving software and hardware in operation, *arXiv preprint arXiv:2309.03102*, 2023.

[20] G. Li, Y. Li, S. Jha, T. Tsai, M. Sullivan, S.K.S. Hari, Z. Kalbarczyk, R. Iyer, AV-FUZZER: Finding safety violations in autonomous driving systems, *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, Coimbra, Portugal, 2020, pp. 25-36, https://doi.org/10.1109/ISSRE5003.2020.00012.

[21] D.L. Bergin, Cyber-attack and defense simulation framework, *Journal of Defense Modeling and Simulation* 12 (4) (2015) 383–392, https://doi.org/10.1177/1548512915593528.

[22] I. Durlik, T. Miller, E. Kostecka, Z. Zwierzewicz, A. Łobodzińska, Cybersecurity in autonomous vehicles—Are we ready for the challenge?, *Electronics* 2024, 13, 832, https://doi.org/10.3390/electronics13132654.

[23] I. Rasheed, F. Hu, L. Zhang, Deep reinforcement learning approach for autonomous vehicle systems for maintaining security and safety using LSTM-GAN, *Computer Communications* 188 (2022) 68–78, https://doi.org/10.1016/j.vehcom.2020.100266.

[24] T. Yasin Rezapour, E. Zeinali, R. Bbrahimi Atani, M. M. Gilanian Sadeghi, A Novel Epidemic-Inspired Routing Protocol for Internet of Vehicles Leveraging the Covid-19 Spread Model, *Journal of Computing and Security*, Vol. 12, No. 1, pages 85-100, https://doi.org/2025,10.22108/jcs.2025.144774.1163.

[25] S. Raio, K. Corder, T.W. Parker, G.G. Shearer, J.S. Edwards, M.R. Thogaripally, S.J. Park, F.F. Nelson, Reinforcement learning as a path to autonomous intelligent cyber-defense agents in vehicle platforms, *Sensors* 22 (4) (2022) 1572.

[26] F. Pishdad, R. Ebrahimi Atani, Prevention and detection of botnet attacks in IoT using ensemble learning methods, *Biannual Journal Monadi for Cyberspace Security (AFTA)*, Vol. 13, No. 2, 2024.

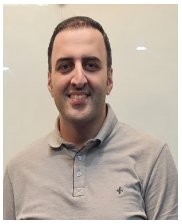
[27] S. Kiruthika, A. Aashin, K. Gopinath, A. Gowtham, Securing connected and autonomous vehicles: Challenges posed by adversarial machine learning and the way forward, *Journal of Physics: Conference Series* 1916 (2021) 012110.

[28] S. Tayeb, M. Pirouz, G. Esguerra, K. Ghobadi, J. Huang, R. Hill, D. Lawson, S. Li, T. Zhan, J. Zhan, S. Latifi, Securing the positioning signals of autonomous vehicles, in: *IEEE International Conference on Big Data (Big Data)*, Boston, MA, USA, 2017, pp. 4522-4528, https://doi.org/10.1109/BigData.2017.8258493.

[29] K. Ren, Q. Wang, C. Wang, Z. Qin, X. Lin, The security of autonomous driving: Threats, defenses, and future directions, *IEEE Communications Magazine* 58 (5) (2020) 74–80,

https://doi.org/10.1109/JPROC.2019.2948775.
[30] Arman Moradi, Mehran Alidoost Nia, and Reza Ebrahimi Atani "Enhancing-Security-and-Safety-in-Autonomous-Vehicles," 2025, GitHub, GitHub repository, https://github.com/ArmanMoradi001/Enhancing-Security-and-Safety-in-Autonomous-Vehicles
[31] Zhongli Wang, Hao Wang, Xin Cui, and Chaochao Zheng, "A Multi-sensing Input and Multi-constraint Reward Mechanism Based Deep Reinforcement Learning Method for Self-driving Policy Learning," School of Electronic Information Engineering, Beijing Jiaotong University, and China Railway Electrification Bureau Group Co., Ltd., Beijing, China, 2020.
[32] Tianmeng Hu, Biao Luo, and Chunhua Yang, "Multi-objective Optimization for Autonomous Driving Strategy Based on Deep Q Network," Accepted: 8 September 2021, Received: 11 August 2021.
[33] Kunkun Hao, Wen Cui, Lu Liu, Yuxi Pan, and Zijiang Yang, "Integrating Data-Driven and Knowledge-Driven Methodologies for Safety-Critical Scenario Generation in Autonomous Vehicle Validation," Research Center of Synkrotron, Xi'an, China, 2020.

Arman Moradi studied Computer Engineering at the University of Guilan in Rasht, Iran. He received his B.Sc. degree in February 2025. During his undergraduate studies, he developed a strong interest in the fields of cybersecurity and artificial intelligence, with a special focus on the intersection of these areas, AI security. His main research interests include cybersecurity, AI, and the design of secure and trustworthy AI systems.



Mehran Alidoost Nia currently is an Assistant Professor in the Faculty of Computer Science and Engineering at Shahid Beheshti University of Tehran, and has a PhD in Software Engineering from the University of Tehran. His research interests include Self-Adaptive and Autonomous Systems, Information Security, Formal Verification, Software Engineering and Programming Languages. His research in PhD program was about probabilistic analysis of self-adaptive and autonomous systems entitled "Runtime Probabilistic Analysis of Self-Adaptive Systems via Formal Approximation Techniques", that was supervised by Mehdi Kargahi (University of Tehran) and Alessandro Abate (University of Oxford). He was a Programme Fellow at the University of York, and collaborated on Assuring Autonomy International Programme (AAIP) from 2020 to 2023.

Reza Ebrahimi Atani Reza Ebrahimi Atani received his B.Sc. degree in Electronics Engineering from the University of Guilan in 2002. He earned his M.Sc. (2004) and Ph.D. (2010) degrees in Electronics Engineering from Iran University of Science and Technology (IUST), where his doctoral research focused on design and secure implementation of symmetric key cryptographic algorithms. During his Ph.D., he conducted research fellowships at FHNW (Switzerland) and KU Leuven (Belgium), specializing in design and secure implementations of stream ciphers. Currently an Associate Professor in the Computer Engineering Department at the University of Guilan, Dr. Atani leads the Computer Security Incident Response Teams (CSIRT) Research Laboratory. His research expertise spans: Design and cryptanalysis of security protocols for wired/wireless networks, AI-driven cybersecurity solutions and threat intelligence, Secure architectures for IoT ecosystems.