Intelligent Obstacle Resilience in Autonomous Vehicles Under Security Threats

Chieh Tsai
Department of Electrical and Computer Engineering
University of Arizona
Tucson, AZ, USA
vegetableclean@arizona.edu

Salim Hariri
Department of Electrical and Computer Engineering
University of Arizona
Tucson, AZ, USA
hariri@arizona.edu

***Abstract*—Autonomous vehicle (AV) security is critical, as even minor software attacks can disrupt key vehicle functions. Resilience, the ability to maintain safe operation despite system compromise is essential. While most existing approaches are tested only in theory or simulation, this paper presents a lightweight, real-time resilience framework implemented on the physical QCar 2 and virtual platform from Quanser, using the NVIDIA Jetson AGX Orin as the main Electronic Control Unit (ECU) for perception and decision-making. We conduct a CVSS-inspired threat analysis, focusing on a configuration tampering attack targeting the main ECU. Through extensive analysis and experiments measuring detection delay and success rate, we demonstrate that this attack disrupts object detection and braking functions. In addition, our resilience framework effectively restores vehicle operation, improving the Mission Success Rate from 0% to over 90% with minimal overhead. This work advances practical autonomous vehicle resilience and lays a solid foundation for future developments in over-the-air resilient updates and fleet diagnostics.**

***Index Terms*—Autonomous Vehicles, Software-based Attacks, Electronic Control Unit, Resilience, CVSS analysis, QCar Platform, Advanced Driver-Assistance Systems, Mission Success Rate.**

I. Introduction

Autonomous Vehicles (AVs), as cyber-physical systems, face increasing threats due to both internal complexity and external environmental uncertainties [1]–[3]. Their core functions—perception, decision making, and control—rely on tightly integrated systems and multiple sensors that communicate through protocols such as CAN and Ethernet. The CAN bus and automotive Ethernet, while critical to inter-module communication, lack strong built-in security and are vulnerable to injection, spoofing, and denial-of-service attacks [4]. Any malfunction or compromise within these interconnected modules can pose significant safety risks.

Emerging studies have shown that malicious actors are actively exploiting these vulnerabilities. For example, LiDAR spoofing [5], GPS falsification [6], and camera-based misclassification attacks [7] have each been demonstrated to undermine vehicle perception and navigation accuracy. Meanwhile, perception models and in-vehicle control software face growing risks from software-based threats such as adversarial inputs, data poisoning, and malicious code injection [8].

While AVs often incorporate baseline safety and fault-tolerance mechanisms, these systems are not designed to actively defend against intentional and adaptive threats [9]. Hardening against known failures alone is insufficient. Instead, resilience—defined as the ability of a system to maintain essential functionality even when partially compromised—has emerged as a critical goal in dependable AV design [10].

Despite growing interest, most proposed resilience strategies remain at the simulation stage, limited by the high cost and risk of real-world testing [11]. In this paper, we focus on software-based attacks, particularly those targeting perception components through parameter manipulation or configuration tampering. These attacks are particularly insidious as they bypass physical sensor protections and remain hidden within the system, often without raising alerts. To address these challenges, we leverage the QCar 2 platform—a hybrid system that supports both physical and virtual testing—to develop and evaluate lightweight, real-time resilience mechanisms. The main contributions of this paper are as follows:

- **Identifying real-world threats:** We identify and demonstrate software-based security threats on a real autonomous vehicle platform, the Quanser QCar2, and propose mitigation strategies.
- **Developing a lightweight resilience method:** We focus on a configuration tampering attack and design a hash-based resilience method with a probability model to detect and respond to abnormal behavior.
- **Implementation and evaluation:** We implement our method on both the simulated and physical QCar platforms to show its effectiveness and highlight the benefits of testing with real hardware.

At the end of this introduction, we briefly summarize the structure of the paper. The rest of the paper is organized as follows: Section II reviews related work, Section III describes the problem and resilience design, Section IV presents experimental results, and Section V concludes the paper.

II. Related Work

A. Software-Based Attacks in Autonomous Vehicles

Software-based threats in AVs commonly target the internal perception pipeline, control stack, or runtime environment.

Unlike physical-layer attacks that manipulate sensor input externally, software attacks compromise system behavior from within. Examples include adversarial perturbations targeting machine learning models [8], [12], poisoning during training phases, or malicious configuration changes that suppress object detection. In addition, ECUs, which coordinate vehicle functions, are vulnerable to runtime tampering and code injection [13]. These threats can bypass physical safeguards and propagate silently through the system, making them particularly dangerous [14].

B. Resilience Mechanisms and System Recovery

Resilience engineering in autonomous vehicles ensures critical functions remain operational under any type of compromises Frameworks like N-Version Programming enhance resilience through redundancy and voting across diverse components [10]. However, both attack and defense mechanisms face practical challenges. Realistic software-level attacks require precise control and safety measures. For instance, adversarial attacks against YOLO-based perception in AVs expose vulnerabilities to spoofing, adversarial inputs, and DoS [15], but are typically evaluated in simulations like CARLA [16], highlighting difficulties in physical deployment. Meanwhile, validating resilience strategies on real platforms is limited by cost, integration, and safety constraints. Many methods remain simulation-bound with limited real-time implementation [10], [17]. Compared with prior approaches that remain largely simulation-based, our work provides a unique contribution through physical validation on the QCar2 platform. While direct side-by-side comparisons with clickjacking or other cyberattack detection tools are challenging due to differing platforms and definitions, our resilience metrics (MSR, VPK, and TTV) enable fair quantitative evaluation of defense effectiveness.

III. Software Threats in AV: Problem and Resilience Design

This section begins by analyzing the architecture of our autonomous vehicle system and formulating the resilience problem. We then conduct a quantitative threat analysis to evaluate software-based vulnerabilities across key modules. Based on the results, we select a representative attack case for in-depth examination and develop a resilience method to defend against it. While the design is driven by a specific scenario, the proposed protection strategy is generalizable to a broader class of software threats.

A. System Model and Its Potential Vulnerabilities

The NVIDIA Jetson AGX Orin serves as the primary Electronic Control Unit (ECU) in our system, responsible for key autonomous vehicle functions. Following the system model described in [18], [19], our architecture consists of three main modules: *Perception*, *Decision Making*, and *Control*, as illustrated in Fig. 1.

Perception Module: This layer is divided into two functional streams:

- *Obstacle Avoidance Stream:* RGB and depth camera inputs are processed by a YOLO-Seg segmentation model [20] to detect and classify obstacles. Subsequently, obstacle localization and ranging are performed to compute object distances using depth data. This stream outputs the set of objects $\mathcal{O}_t = \{(b_i, c_i, s_i, d_i)\}$, where b_i is the bounding box, c_i the class label, s_i the confidence score, and d_i the estimated distance. A false negative (e.g., missed detection of a pedestrian) or misclassification (e.g., stop sign relabeled) can lead to catastrophic outcomes.
- *Trajectory Planning Stream:* Data from the LiDAR (GPS), gyroscope, and tachometer are fused via an Extended Kalman Filter (EKF) [21] to track vehicle pose and velocity. This feeds into a path planning module that determines safe, goal-directed trajectories under the current vehicle state. Though important for global navigation, this stream assumes obstacle input from the YOLO-based obstacle detection, and therefore inherits its vulnerabilities.

Decision Making Module: This module interprets signals from both planning and obstacle detection. It makes high-level decisions making (e.g., stop, go, turn) based on detected objects and planned trajectories. Any misguidance from upstream—particularly from tampered detection output—can lead to incorrect decisions.

Control Module: This module executes the driving policy by translating high-level decisions into low-level commands, including steering angle and vehicle speed. Although deterministic, it is inherently dependent on upstream accuracy and thus susceptible to silent failure propagation.

B. Problem Formulation

We model the autonomous vehicle as a pipeline with three stages:

$$I_t \xrightarrow{f(\theta)} O_t \xrightarrow{g} D_t \xrightarrow{h} C_t$$

where I_t is sensory input, θ are system parameters, O_t is the perception output, D_t is the driving decision, and C_t is the control command.

We assume the attacker can modify internal parameters θ, such as detection thresholds or class mappings, without access to the physical sensors. These changes may distort $O_t = \{(b_i, c_i, s_i, d_i)\}$, leading to incorrect decisions and unsafe control actions.

C. Threat Analysis

To assess the severity of software-based attacks in autonomous vehicles, we adopt the Common Vulnerability Scoring System (CVSS) v3.1, [22] a widely recognized framework for evaluating the risk of cyber threats based on exploitability and impact.

Drawing on prior studies of code tampering, denial-of-service (DoS), and spoofing attacks [23], [24], we identify a set of realistic software-based attack scenarios targeting the perception, decision-making, and control modules. These

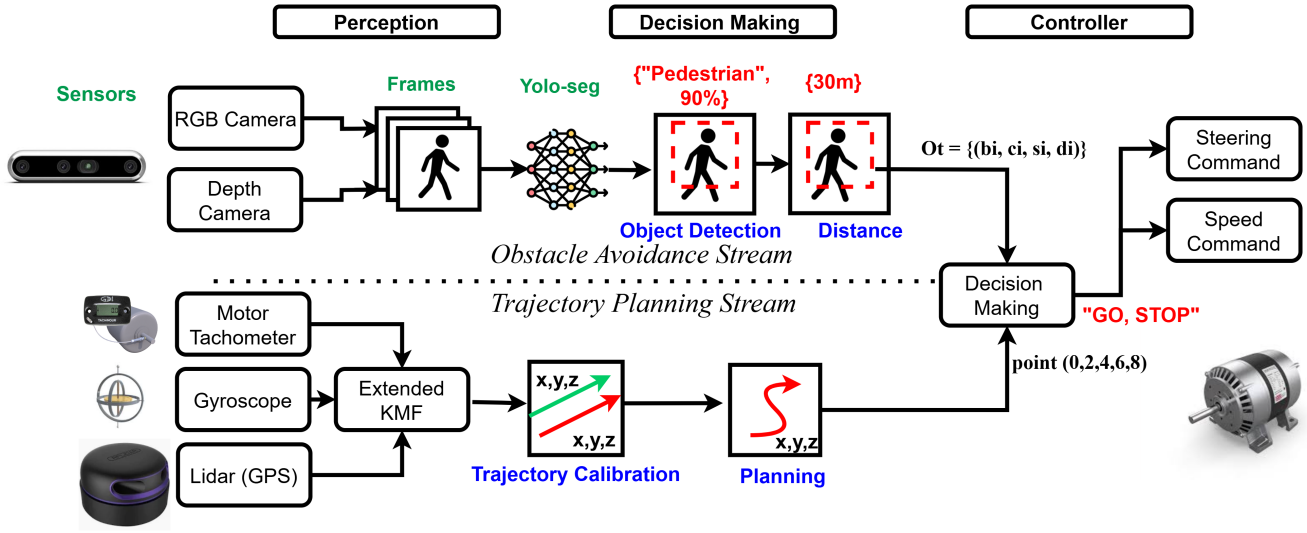


Fig. 1: System architecture illustrating the flow from perception to control in autonomous vehicles.

include adversarial manipulation of model parameters, ADAS-like ECU-level configuration tampering (e.g., threshold adjustments or runtime patching), and sensor spoofing—each simulated and executed on the Jetson-based perception and decision module to reflect realistic AV vulnerabilities. To evaluate the impact of each attack, we adopt a CVSS-inspired metric to quantify severity across impact, exploitability, and stealthiness dimensions.

- **YOLO Threshold Tampering:** Reduces detection threshold to suppress low-confidence objects. **Score: 9.0 (Critical)**
- **Class Mapping Swap:** Reassigns class labels (e.g., stop sign to speed limit), causing critical misclassification. **Score: 6.7 (Medium)**
- **Depth Mask Injection:** Injects falsified depth input to disrupt distance estimation. **Score: 7.2 (High)**
- **LiDAR Replay Spoofing:** Uses recorded data to mislead localization systems. **Score: 6.4 (High)**
- **IMU Yaw Drift Injection:** Slowly injects drift into yaw readings, degrading long-term pose estimation. **Score: 2.8 (Low)**
- **Velocity Signal Freezing:** Freezes velocity data stream, misleading planning modules. **Score: 7.1 (High)**
- **Policy Hijack via Config Patch:** Alters planning policy to ignore critical cues (e.g., stop signs). **Score: 6.7 (Medium)**
- **Cost Function Manipulation:** Adjusts planning weights, skewing path safety. **Score: 5.3 (Medium)**
- **Steering Control Bias Injection:** Introduces a bias to the control signal without validation. **Score: 5.3 (Medium)**

Each score is calculated according to CVSS v3.1, based on exploitability metrics (access vector, complexity, privileges) and impact metrics (confidentiality, integrity, availability). As shown in Table I, most high-severity attacks are concentrated in the **Perception module**, particularly those affecting object detection and spatial awareness. This is especially critical in the context of **obstacle avoidance**, where timely and accurate perception of the environment directly influences vehicle safety. Compared to Planning and Control modules, perception-layer vulnerabilities tend to require lower privileges while yielding higher impact on system behavior.

Given its pivotal role and elevated risk profile, the next section focuses on analyzing the resilience of the perception pipeline under software-based attacks, with an emphasis on its implications for safe obstacle avoidance.

D. Case Study: Resilience Design for Perception Tampering Attacks

Based on the threat scenarios outlined earlier, we focus on a representative attack usecase targeting the primary ADAS-like ECU, specifically the NVIDIA Jetson AGX Orin, which handles perception and decision-making tasks. The attack involves software-level tampering within this ECU, such as manipulating YOLO detection thresholds or remapping class labels. Although these modifications may appear minor in code, they can severely degrade system performance by bypassing safety-critical object detection mechanisms without triggering conventional sensor-level alarms.

To address this, we design a generalizable resilience framework that begins with this specific usecase and can be extended to cover other perception-layer software attacks. The proposed approach introduces a lightweight **hash-based verification module**, which continuously monitors the integrity of key perception components—such as model weights, configuration files, and execution scripts. At runtime, cryptographic hashes (SHA-256) [25] are computed periodically and compared with pre-verified references. When a mismatch is detected, the system switches execution to a secure **backup module** with known-good behavior, ensuring continuity and correctness, shown in Table II

This case-driven resilience design is validated using a stop sign suppression attack—caused by class remapping—as

TABLE I: CVSS v3.1 Scores for Software-Based Attacks by Module

Threat Name	Module	AV	AC	PR	UI	S	C	I	A	Score	Severity
YOLO Threshold Tampering	Perception	L	L	N	N	C	N	H	H	9.0	Critical
Class Mapping Swap	Perception	L	H	N	N	C	N	H	L	6.7	Medium
Depth Mask Injection	Perception	L	H	H	N	C	N	H	H	7.2	High
LiDAR Replay Spoofing	Perception	L	H	L	N	C	N	H	L	6.4	High
IMU Yaw Drift Injection	Perception	L	H	L	N	C	N	L	N	2.8	Low
Velocity Signal Freezing	Control	L	L	N	N	C	N	N	H	7.1	High
Policy Hijack via Config Patch	Planning	L	L	H	N	C	N	H	L	6.7	Medium
Cost Function Manipulation	Planning	L	H	H	N	C	N	H	N	5.3	Medium
Steering Control Bias Injection	Control	L	H	H	N	C	N	H	N	5.3	Medium

Abbreviations: AV – Attack Vector, AC – Attack Complexity, PR – Privileges Required, UI – User Interaction, S – Scope, C – Confidentiality, I – Integrity, A – Availability.
L – Low, H – High, N – None, C – Changed.

TABLE II: Hash-Based Integrity Protection Mechanism

Attack Detection Mechanism	Attack Mitigation	Attack Response
Hash-based integrity checking detects mismatch between expected and actual hash values	Switch to verified backup module with trusted parameters	Trigger alarm, log event, and notify controller for safe fallback

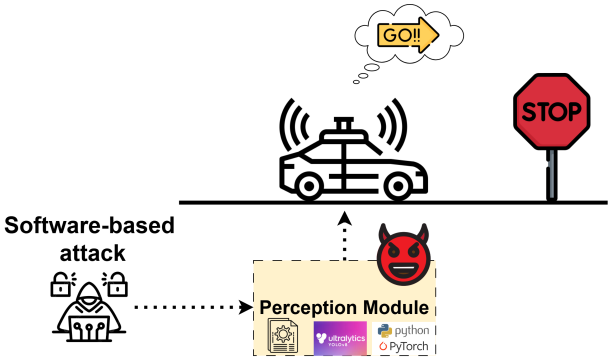


Fig. 2: Example of a software-based perception attack: Stop sign suppression via class index tampering. The vehicle fails to stop at the intersection.

shown in Fig.2. The proposed architecture (Fig.3) effectively mitigates such tampering by continuously verifying runtime integrity and dynamically switching to a trusted backup module when anomalies are detected. Specifically, the system executes the primary module under normal conditions (i.e., when the hash validation succeeds), and falls back to a pre-verified module when discrepancies are detected.

Importantly, this resilience strategy extends beyond stop signs. We analyzed multiple high-priority object types and designed adaptive responses based on both detection confidence and environmental context, ensuring appropriate behavior under a wide range of attack scenarios.

Importantly, this resilience strategy generalizes beyond sign scenarios. We analyzed multiple critical object types and designed tailored responses based on detection confidence and context, enabling appropriate reactions under different attack scenarios.

Table III outlines the expected vehicle behavior under protection mechanisms for each object type Fig. 4, including stop signs, pedestrian, traffic lights, and vehicles. The following tables detail the detection conditions and corresponding recovery mechanisms activated under various tampering scenarios.

E. Resilience Probability Model

Inspired by the broad cyber resilience framework in [26], we propose a refined probabilistic model that incorporates detailed system dynamics and practical attacker assumptions, enabling a more precise evaluation of our system's ability to maintain correct behavior under software-based perception attacks

We define the overall resilience probability as:

$$P_{\text{res}} = (1 - P_{\text{atk}}) + P_{\text{atk}} \cdot P_{\text{def}} \tag{1}$$

Where:

- P_{atk}: Probability that a software-based perception attack occurs during a given mission segment.
- P_{def}: Probability that the resilience mechanism successfully detects and mitigates the attack in time.

We further decompose P_{def} as:

$$P_{\text{def}} = \eta \cdot e^{-\lambda t_d} \cdot (1 - \mu v_0) \cdot \delta(O, A) \tag{2}$$

Where:

- η: Base effectiveness of the hash-checking mechanism under ideal conditions.
- t_d: Total delay from attack onset to hash mismatch detection.
- v_0: Vehicle speed at the time of attack; higher speeds reduce reaction time.
- λ: Sensitivity coefficient quantifying how detection delay degrades defense effectiveness.
- μ: Sensitivity coefficient describing the impact of vehicle speed on available reaction window.
- $\delta(O, A) \in [0, 1]$: Contextual difficulty factor based on the criticality of object O and attack type A.

This formulation highlights that:

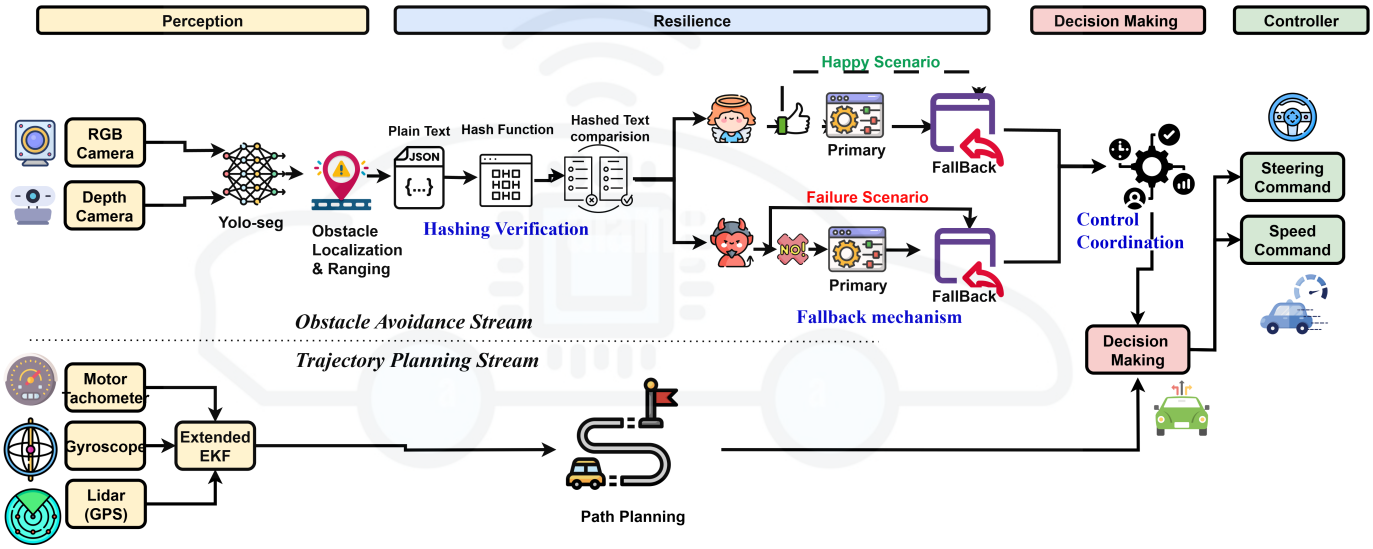


Fig. 3: System-level resilience architecture.

TABLE III: Object Type and Car Response (with Defense)

Object Type	**Car Response (with Defense)**
Stop Sign	Car comes to a full stop, waits briefly, then resumes based on contextual judgment.
Traffic Light	Car adjusts speed based on detected light signal.
Vehicle	Maintains safe following distance or stops if vehicle ahead is lost.
Pedestrian	Car halts ahead of the person and waits until the path is clear.

Fig. 4: Key object types considered in the resilience design: stop sign, pedestrian, vehicle ahead, and traffic light.

- Increasing vehicle speed ($v_0 \uparrow$) reduces defense success due to shorter safe margins.
- Higher detection delays ($t_d \uparrow$) exponentially degrade defense capability.
- Attacks targeting high-priority objects or stealthier attack vectors lower $\delta(O, A)$, making defense harder.

Notably, this model refines and extends the cyber resilience solution presented in [26], where the malware and bonware parameters (M and B) conceptually correspond to the decay factor and the recovery factor in our framework.

IV. Experimental Results

In this section, we evaluate the effectiveness of our resilience mechanism. The following subsections describe the system setup for both the physical car and the virtual simulation, followed by the evaluation results.

A. System Setup

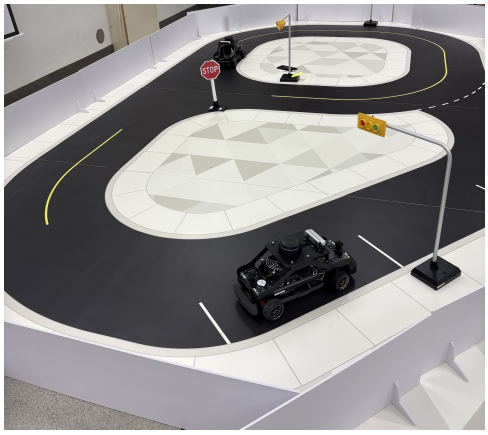
The vehicle used in our experiments is the **QCar 2** developed by **Quanser Inc. (Canada)**. QCar 2 is a 1/10th scale, open-architecture self-driving vehicle designed for academic use and research. It is the primary platform in Quanser's *Self-Driving Car Studio*, featuring a high-performance **NVIDIA Orin AGX** processor and equipped with a comprehensive suite of **inertial, visual, and ranging sensors**.

QCar 2 supports both physical and virtual testing environments, Please refer to Figure 5 for more details:

- The **physical QCar 2** and the large roadmap used in the studio are scaled down to **1:10** of real-world dimensions.
- The **virtual QCar 2** operates in a **1:1 scale** environment and behaves like a full-sized real vehicle.

This testbed includes additional infrastructure elements such as **stop signs**, **traffic lights**, and **static vehicles and**

pedestrian to simulate real-world driving conditions. These objects are positioned strategically across the test map to create perception-driven control scenarios. Both the physical and virtual maps are equipped with camera-friendly lighting and GPS simulation sources to support consistent evaluation.



(a) Physical QCar 2 with 1:10 scaled map.

(b) Virtual QCar 2 in 1:1 scale simulation.

Fig. 5: Physical and virtual QCar 2 setups.

We apply a scale factor of 10, meaning that speeds in the physical QCar 2 are one-tenth of their real-world equivalents. For example, 1.2m/s in QCar represents 12m/s (43.2km/h) in real-life driving. Please refer to Table IV for more details.
The Physical QCar 2 typically runs at approximately **10 frames per second (FPS)** while the virtual QCar 2 can run at **30 frames per second (FPS)** . This setup enables researchers to replicate real-world driving behavior in both physical and simulated environments.

B. Speed Response Over Time

To demonstrate the importance of our resilience method, we conducted experiments comparing system behavior with and without the proposed defense. We first tested a stop sign configuration tampering attack. In this setup, the vehicle speed was set to 0.5m/s, the update rate was 5, and the attack was triggered at approximately 15 seconds.

As shown in Figure 6, when the tampered stop sign appeared, the vehicle equipped with our resilience mechanism successfully detected the anomaly using hash matching and immediately triggered a fallback response. This enabled the vehicle to maintain correct behavior, successfully stopping at subsequent stop signs at around 40s, 50s, and 60s despite the earlier attack. In contrast, without any defense mechanism, the vehicle failed to recognize the tampered stop sign and did not stop as required after 15 seconds of operation. This kind of traffic violation could potentially lead to serious accidents.

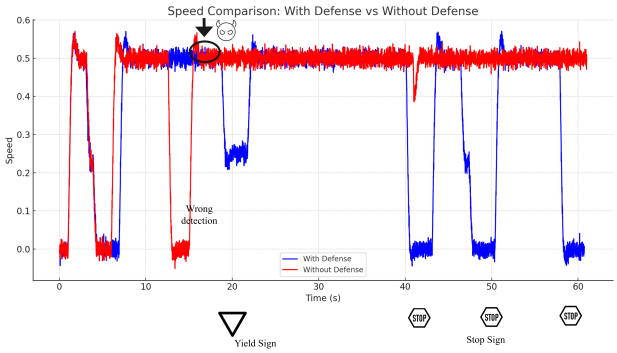


Fig. 6: Vehicle Speed vs. Time under Defense and No Defense

C. Resilience vs. Tampering: Measuring Success Rates Across Different Objects

Building on the previous experiment, we extended our evaluation to various object types to further assess the effectiveness of the proposed resilience mechanism. In this series of tests, the vehicle speed was set to 0.25m/s. Each attack condition was tested five times with a fixed update interval of 5. A mitigation was considered successful if the vehicle responded appropriately to the attack—such as stopping at a stop sign or avoiding a pedestrian.

Our results show that the resilience mechanism performs reliably only up to a speed of 1.8m/s. Beyond this threshold, system performance begins to degrade: only simpler tasks, such as reacting to stop signs, remained consistently successful. Notably, at 1.75m/s, only one test case successfully triggered a pedestrian avoidance response, indicating reduced stability at higher speeds, as shown in Fig. 7.

To address this limitation, we suggest adjusting the update interval in future experiments, as preliminary results indicate a strong correlation between update rate and system stability (see Fig. 8).

D. System Delay-Aware Tuning of Detection Update Rate

Based on our experiments, we observed that factors such as update intervals and time delays—stemming from components like the motor or vehicle speed—play a critical role in overall system performance. Therefore, we evaluated both the system's inherent delays and those introduced by the proposed resilience mechanism, as shown in 9 and 10. Specifically, we measured the following: attack detection time, decision-making time, brake command issuance, actual brake response, and for the resilience mechanism, the time required for attack recognition, fallback execution, and recovery to normal operation.

TABLE IV: Speed Equivalence between QCar 2 and Full-scale Vehicle

QCar 2 Speed (m/s)	Full-scale Speed (m/s)	Speed (mph)	Speed (km/h)
1.2	12	26.8	43.2
1.5	15	33.6	54
4.3 (max)	43	93.6	150

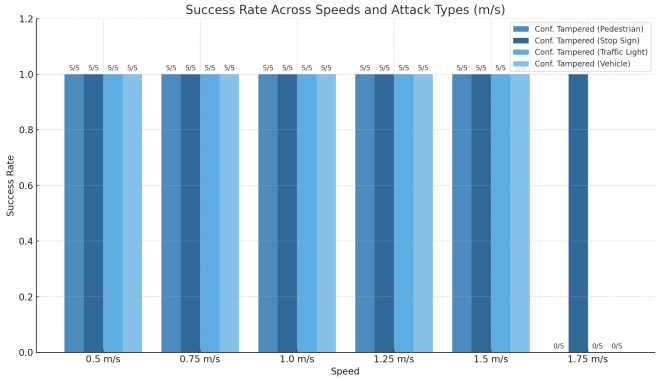


Fig. 7: Success rates across different objects, tested from 0.5 m/s to 1.75 m/s at 0.25 m/s intervals.

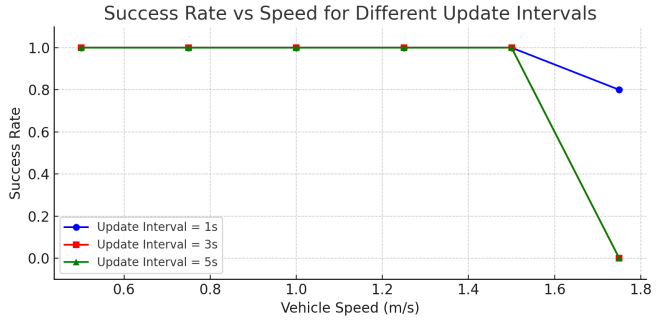


Fig. 8: Comparision different update intervals with pedestrian detection.

Our analysis shows that with the proposed resilience mechanism, the system can detect and recover from attacks in under 0.2 seconds. In our experiments, we triggered the attack at 25 seconds, and the system successfully performed attack detection and fallback recovery within just 0.001 and 0.002 seconds, respectively. In contrast, when the system operates without any defense, it fails to respond appropriately—resulting in missed stop signs or undetected pedestrians. Notably, the resilience mechanism introduces minimal computational overhead, making it a lightweight yet effective solution.

This highlights that while the defense mechanism greatly improves safety, its success depends heavily on the appropriate configuration of update intervals, vehicle speed, and response timing. Careful tuning of these parameters is essential to minimize risk and ensure timely mitigation of threats in real-world scenarios.

E. Real-World Validation on Physical QCar2

To validate our proposed resilience mechanism in a real-world setting, we deployed the system on a physical QCar2 platform. This experiment demonstrates the impact of a parameter tampering attack and the effectiveness of the recovery strategy under real-time constraints.

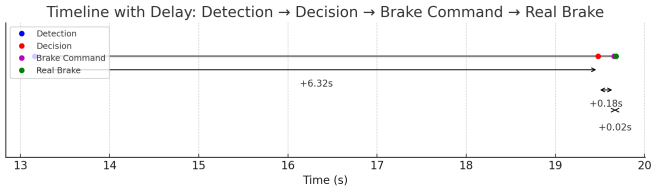


Fig. 9: System Delay Timeline

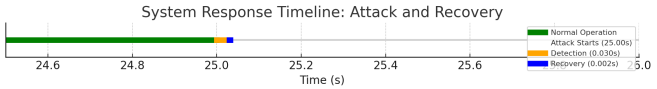


Fig. 10: Detection and Response Timeline (Attack to Braking)

As illustrated in Fig. 11, the attack was simulated by modifying a critical parameter at the 10-second mark: the 'stop sign count' threshold was altered from 5 to 100. This manipulation effectively suppresses the recognition of a stop sign, preventing the vehicle from initiating braking behavior in time.

Without the resilience module, the QCar failed to stop, resulting in a clear violation of the expected behavior. In contrast, our system detected the tampering and triggered the hash-based recovery mechanism. Within a short recovery latency, the correct configuration was restored, enabling the vehicle to recognize the stop sign and apply the brakes successfully.

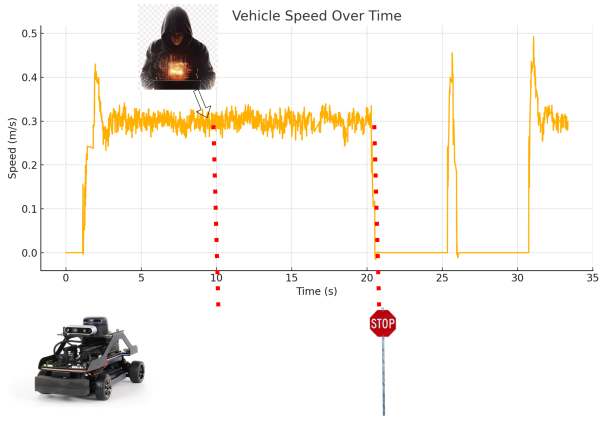


Fig. 11: Real-world vehicle (Qcar2) speed response after tampering and recovery

F. Defense Effectiveness Evaluation via Resilience Metrics

To assess the defense effectiveness, we use three resilience metrics inspired by prior work [27]: Mission Success Rate (MSR), Traffic Violations per Kilometer (VPK), and Time to Traffic Violation (TTV).

• **MSR**: 27 of 30 runs succeeded with defense (90%) versus 0% without. • **VPK**: Violations dropped from 16.67 (no defense) to 5.56 (with defense). • **TTV**: Average violation time increased from 2.1s to stable operation ($>$ 10s).

Table V summarizes the results, showing that our SHA-256-based fallback significantly improves resilience.

TABLE V: Estimated Resilience Metrics Across Defense Conditions

Metric	With Defense	No Defense
Mission Success Rate (MSR)	90%	0%
Traffic Violations per Km (VPK)	5.56	16.67
Time to Traffic Violation (TTV, sec)	$>$ 10 (Stable)	2.1

V. Conclusion

This paper presented a practical framework for evaluating resilience against software-based attacks in autonomous vehicles. Using targeted perception tampering and real-time recovery on the QCar2 platform, we showed that lightweight defenses are both effective and feasible for embedded systems. Our study is limited by dataset size and attack scope, but results across stop signs, pedestrians, vehicles, and traffic lights demonstrate generalizability. Future work will validate the probabilistic model with larger datasets and extend to additional attacks such as LiDAR spoofing and GPS falsification, while developing faster, more accurate resilience methods for real-world deployment.

References

[1] T. Islam, M. A. Sheakh, A. N. Jui, O. Sharif, and M. Z. Hasan, "A review of cyber attacks on sensors and perception systems in autonomous vehicle," *Journal of Economy and Technology*, vol. 1, pp. 242–258, 2023.

[2] Y. Zhu, C. Miao, H. Xue, Y. Yu, L. Su, and C. Qiao, "Malicious attacks against multi-sensor fusion in autonomous driving," in *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, 2024, pp. 436–451.

[3] A. Zaboli, J. Hong, J. Kwon, and J. Moore, "A survey on cyber-physical security of autonomous vehicles using a context awareness method," *IEEE Access*, vol. 11, pp. 136 706–136 725, 2023.

[4] A. B. C. Douss, R. Abassi, and D. Sauveron, "State-of-the-art survey of in-vehicle protocols and automotive ethernet security and vulnerabilities," *Mathematical Biosciences and Engineering*, vol. 20, no. 9, pp. 17 057–17 095, 2023.

[5] T. Sato, R. Suzuki, Y. Hayakawa, K. Ikeda, O. Sako, R. Nagata, R. Yoshida, Q. A. Chen, and K. Yoshioka, "On the realism of lidar spoofing attacks against autonomous driving vehicle at high speed and long distance," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2025.

[6] M. Mehrab Abrar, R. Islam, S. Satam, S. Shao, S. Hariri, and P. Satam, "Gps-ids: An anomaly-based gps spoofing attack detection framework for autonomous vehicles," *arXiv e-prints*, pp. arXiv–2405, 2024.

[7] B. Nassi, Y. Mirsky, D. Nassi, R. Ben-Netanel, O. Drokin, and Y. Elovici, "Phantom of the adas: Securing advanced driver-assistance systems from split-second phantom attacks," in *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, 2020, pp. 293–308.

[8] H.-Y. Lin and B. Biggio, "Adversarial machine learning: Attacks from laboratories to the real world," *Computer*, vol. 54, no. 5, pp. 56–60, 2021.

[9] G. Lippi, M. Aljawarneh, Q. Al-Na'amneh, R. Hazaymih, L. D. Dhomeja *et al.*, "Security and privacy challenges and solutions in autonomous driving systems: A comprehensive review," *Journal of Cyber Security and Risk Auditing*, vol. 2025, no. 3, pp. 23–41, 2025.

[10] J. M. Qurashi, K. Jambi, F. Alsolami, F. E. Eassa, M. Khemakhem, and A. Basuhail, "Resilient countermeasures against cyber-attacks on self-driving car architecture," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 11, pp. 11 514–11 543, 2023.

[11] O. O. Ajayi, A. S. Adebayo, and N. Chukwurah, "Addressing security vulnerabilities in autonomous vehicles through resilient frameworks and robust cyber defense systems," 2025.

[12] A. D. M. Ibrahum, M. Hussain, and J.-E. Hong, "Deep learning adversarial attacks and defenses in autonomous vehicles: a systematic literature review from a safety perspective," *Artificial Intelligence Review*, vol. 58, no. 1, pp. 1–53, 2025.

[13] I. Pekaric, C. Sauerwein, S. Haselwanter, and M. Felderer, "A taxonomy of attack mechanisms in the automotive domain," *Computer Standards & Interfaces*, vol. 78, p. 103539, 2021.

[14] Z. Ju, H. Zhang, X. Li, X. Chen, J. Han, and M. Yang, "A survey on attack detection and resilience for connected and automated vehicles: From vehicle dynamics and control perspective," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 4, pp. 815–837, 2022.

[15] J. Im Choi and Q. Tian, "Adversarial attack and defense of yolo detectors in autonomous driving scenarios," in *2022 IEEE intelligent vehicles symposium (IV)*. IEEE, 2022, pp. 1011–1017.

[16] R. Li, S. Dai, and A. Xiong, "Adversarial object-evasion attack detection in autonomous driving contexts: A simulation-based investigation using yolo." ISOC, 2024.

[17] H. Alemayehu and A. Sargolzaei, "Testing and verification of connected and autonomous vehicles: A review," *Electronics*, vol. 14, no. 3, p. 600, 2025.

[18] M. Sivakumar, A. B. Belle, J. Shan, O. Odu, and M. Yuan, "Design of the safety case of the reinforcement learning-enabled component of a quanser autonomous vehicle," in *2024 IEEE 32nd International Requirements Engineering Conference Workshops (REW)*. IEEE, 2024, pp. 57–67.

[19] Quanser Inc., "Qcar 2: Self-driving car for education and research," https://www.quanser.com/products/qcar/, n.d., accessed: 2025-06-11.

[20] U. Contributors, "Ultralytics yolov8 segment models," *Ultralytics Docs*, 2024, accessed: 2025-06-11.

[21] P. S. Maybeck, *Stochastic Models, Estimation, and Control*. Academic Press, 1979, vol. 1.

[22] FIRST.Org, Inc., *Common Vulnerability Scoring System (CVSS) Version 3.1 Specification Document*, FIRST.Org, Inc., USA, Jun. 2019, latest official version, with calculator and user guide available online. [Online]. Available: https://www.first.org/cvss/v3.1/specification-document

[23] J. M. Qurashi, K. M. Jambi, F. E. Eassa, M. Khemakhem, F. Alsolami, and A. A. Basuhail, "Toward attack modeling technique addressing resilience in self-driving car," *IEEE Access*, vol. 11, pp. 2652–2673, 2022.

[24] B. B. Madan, M. Banik, and D. Bein, "Securing unmanned autonomous systems from cyber threats," *The Journal of Defense Modeling and Simulation*, vol. 16, no. 2, pp. 119–136, 2019.

[25] N. I. of Standards and T. (NIST), "Secure hash standard (shs)," U.S. Department of Commerce, Federal Information Processing Standards Publication FIPS PUB 180-4, 2015, accessed: 2025-06-11. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf

[26] A. Kott, M. J. Weisman, and J. Vandekerckhove, "Mathematical modeling of cyber resilience," in *MILCOM 2022-2022 IEEE Military Communications Conference (MILCOM)*. IEEE, 2022, pp. 849–854.

[27] S. Jha, S. S. Banerjee, J. Cyriac, Z. T. Kalbarczyk, and R. K. Iyer, "Avfi: Fault injection for autonomous vehicles," in *2018 48th annual ieee/ifip international conference on dependable systems and networks workshops (dsn-w)*. IEEE, 2018, pp. 55–56.