DOCTORAL THESIS

Cybersecurity Testing and Attack Propagation Analysis of Autonomous Driving Software

Andrew James Roberts

TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY
TALLINN 2025

TALLINN UNIVERSITY OF TECHNOLOGY
DOCTORAL THESIS
41/2025

Cybersecurity Testing and Attack Propagation Analysis of Autonomous Driving Software

ANDREW JAMES ROBERTS

TAL TECH PRESS

TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies
Department of Software Science

The dissertation was accepted for the defence of the degree of Doctor of Philosophy on 27th May 2025

Supervisor: Professor Dr. Olaf Manuel Maennel
School of Computer and Mathematical Sciences
University of Adelaide
Adelaide, Australia

Co-supervisors: Dr.-Ing. Mohammad Hamad
Department of Computer Engineering
Technical University of Munich
Munich, Germany

Assoc. Professor Dr. Raivo Sell
Department of Mechanical and Industrial Engineering, School of Engineering
Tallinn University of Technology
Tallinn, Estonia

Opponents: Professor Dr. Iain Phillips
Loughborough University
Loughborough, United Kingdom

Asst. Professor Dr. Monowar Hasan
Washington State University
Washington, United States of America

Defence of the thesis: 25th June 2025, Tallinn

Declaration:
Hereby I declare that this doctoral thesis, my original investigation and achievement, submitted for the doctoral degree at Tallinn University of Technology, has not been submitted for any academic degree elsewhere.

Andrew James Roberts

signature

Copyright: Andrew James Roberts, 2025
ISSN 2585-6898 (publication)
ISBN 978-9916-80-317-2 (publication)
ISSN 2585-6901 (PDF)
ISBN 978-9916-80-318-9 (PDF)
DOI https://doi.org/10.23658/taltech.41/2025
Printed by Koopia Niini & Rauam

Roberts, A. J. (2025). *Cybersecurity Testing and Attack Propagation Analysis of Autonomous Driving Software* [TalTech Press]. https://doi.org/10.23658/taltech.41/2025

TALLINNA TEHNIKAÜLIKOOL
DOKTORITÖÖ
41/2025

Autonoomse sõiduki juhtimistarkvara küberturvalisuse testimine ja rünnakute leviku analüüs

ANDREW JAMES ROBERTS

Contents

List of Publications

The present Ph.D. thesis is based on the following publications that are referred to in the text by Roman numbers.

I A. Roberts, L. Teply, M. Bellone, M. Pese, O. Maennel, M. Hamad, and S. Steinhorst. Fuzzsense: Towards a modular fuzzing framework for autonomous driving software. In *arXiv*, 2025

II A. Roberts, M. Malayjerdi, M. Bellone, R. Sell, O. Maennel, M. Hamad, and S. Steinhorst. Adsecdata platform: An open-source data platform for autonomous driving cybersecurity. In *2025 IEEE 101st Vehicular Technology Conference (VTC2025-Spring)*, pages 1–7, 2025

III A. Roberts, J. Cheng, O. Maennel, M. Hamad, and S. Steinhorst. Adseclang: A domain-specific language for cybersecurity testing of autonomous vehicles. In *2025 IEEE 101st Vehicular Technology Conference (VTC2025-Spring)*, pages 1–6, 2025

IV A. Roberts, M. Malayjerdi, M. Bellone, R. Sell, O. Maennel, M. Hamad, and S. Steinhorst. Analysis of autonomous driving software to low-level sensor cyber attacks. In *2025 IEEE/ACM 20th Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 1–11, 2025

V M. Hamad, A. Finkenzeller, M. Kühr, A. Roberts, O. Maennel, V. Prevelakis, and S. Steinhorst. React: Autonomous intrusion response system for intelligent vehicles. *Computers & Security*, 145:104008, 2024

VI A. Roberts, M. R. H. Iman, M. Bellone, T. Ghasempouri, J. Raik, O. Maennel, M. Hamad, and S. Steinhorst. Adassure: Debugging methodology for autonomous driving control algorithms. In *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6, 2024

VII A. Roberts, M. Malayjerdi, M. Bellone, O. Maennel, and E. Malayjerdi. Analysing adversarial threats to rule-based local-planning algorithms for autonomous driving. *Inaugural Symposium on Vehicle Security and Privacy (VehicleSec 2023) with NDSS*, pages 1–8, 2023

VIII A. Roberts, S. Marksteiner, M. Soyturk, B. Yaman, and Y. Yang. A global survey of standardization and industry practices of automotive cybersecurity validation and verification testing processes and tools. *SAE International Journal of Connected and Automated Vehicles*, 7, 11 2023

IX M. Malayjerdi, A. Roberts, O. M. Maennel, and E. Malayjerdi. Combined safety and cybersecurity testing methodology for autonomous driving algorithms. *Proceedings of the 6th ACM Computer Science in Cars Symposium*, pages 1–10, 2022

X A. Roberts, O. Maennel, and N. Snetkov. Cybersecurity test range for autonomous vehicle shuttles. *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*, pages 239–248, 2021

Author’s Contributions to the Publications

I In **Publication I** as the first author, I designed the FuzzSense architecture. The FuzzSense architecture that I designed furthers state-of-the-art through targeting the AD sensor and enables fuzzing multiple targets using diverse fuzzing techniques. FuzzSense is an initial contribution to the community of an extensible fuzzing architecture that will enable the community to test AD systems in a more agile and repeatable manner. The results of the initial testing of reveal its ability to find vulnerabilities in state-of-the-art AD software frameworks. I authored the manuscript

II In **Publication II** as the first author, I designed the framework of the ADSecData platform. ADSecData is the first contribution in the research community that provides analysis of the utility of AD data for cybersecurity. ADSecData is also the only open-source AD cybersecurity data platform. I conducted the collection and analysis of available automotive cybersecurity datasets. I highlighted the research gaps in the community and then I provide the ADSecData data generation process. The results of the paper is available, open-source AD cybersecurity datasets. I authored the manuscript.

III In **Publication III** as the first author, I designed ADSecLang. ADSecLang is a domain-specific language for AD cybersecurity. I defined the taxonomy and workflows and analysis of the experiments. The ADSecLang validated performance in discovery vulnerabilities in state-of-the-art end-to-end AD software. ADSecLang is the first contribution for the development of structured cybersecurity testing of AD systems. I authored the manuscript.

IV In **Publication IV** as the first author, I originated the idea of analysis of the propagation of attacks to AD systems. I conducted a literature review of real-world and academic research on attacks to automotive and aviation sensors. I developed the attack model and conducted the propagation analysis. The propagation analysis is one of the first contributions to the research community of analysis of the effect of cyber attacks to the integrated hardware/software architecture in AD systems. I authored the manuscript.

V In **Publication V** as a contributing author, I provided input to the design of intrusion response in automotive systems. I analysed the results of the technical experiments and provided feedback in collaboration with other authors. I co-authored the manuscript.

VI In **Publication VI** I was the joint first author. I originated the idea of the ADAssure methodology. I provided the novel approach of using variances in vehicle dynamics data as a means of fingerprinting cyber attack behaviour. I contributed the datasets and conducted the manual analysis to compare the results of the automated analysis and fine-tune the assertions which are used as deterministic rules on the control system. I co-authored the manuscript.

VII In **Publication VII** as the first author, I developed the attack model which consists of 3 diverse attacks (lateral and longitudinal deviation and time-message delay) to the cost-based planning algorithm. I conducted the analysis of the results which demonstrated that these attacks can successfully exploit vulnerabilities in the planning software of the case study vehicle (iseAuto). I authored the manuscript.

VIII In **Publication VIII**, as the joint first author, I wrote the sections on automotive cybersecurity regulation, academic survey and analysis of processes and tools used in industry. This work was the first publication on automotive cybersecurity by the International Alliance for Mobility Testing and Standardisation Working Group 4, Cybersecurity, for which I was a leading contributor. It is also one of the first studies on automotive cybersecurity regulations, standards and industry tools/processes. I co-authored the manuscript.

IX In **Publication IX** as the joint first author, I provided the novel approach of combining safety and security metrics in a methodology to enhance analysis of cybersecurity testing. I developed the attack model (adversarial LiDAR point cloud injection) and the metrics for security. The results validated that the attack model was able to exploit vulnerabilities in the perception and planning modules of the case study vehicle (iseAuto). I conducted further analysis of the testing results and collaborated with the software developers to provide recommendations for improving AD software robustness. I co-authored the manuscript.

X In **Publication X** as the first author, I conducted an analysis of AD testbeds and AD programs operating in real-world conditions. The analysis resulted in the choice of DuckieTown for further evaluation. I collaborated with the DuckieTown Foundation, procured the hardware and built the DuckieTown testbed at TalTech. I developed the cyber threat generation method, which involved interviewing real-world AD operators to ascertain realistic threat scenarios. I conducted the experiments and the analysis which validated the ability of the small-factor cyber-physical testbed to support edge and corner cases for AD cybersecurity. I authored the manuscript.

Abbreviations

ABS	Anti-Lock Braking
ACC	Advanced Cruise Control
AD	Autonomous Driving
ADAS	Advanced Driver Assistance System
AI	Artificial Intelligence
AV	Autonomous Vehicle
CAN	Controller Area Network
CNN	Convolutional Neural Network
DNN	Deep Neural Network
DTC	Distance-to-Collision
ECU	Electronic Control Unit
E/E	Electrical/Electronic Components
EMI	Electromagnetic Interference
ESC	Electronic Stability Control
GNSS	Global Navigational Satellite System
GPS	Global Positioning System
HiL	Hardware-in-the-Loop
IMU	Inertial Measurement Unit
IRS	Intrusion Response System
IDS	Intrusion Detection System
LKAS	Lane-Keeping Assistance System
LiDAR	Light Detection and Ranging
LIN	Local Interconnect Network
MOST	Media Oriented Systems Transport
NDT	Normal Distributions Transform
NPC	Non-Playable Character
ODD	Operational Design Domain
PID	Proportional–Integral–Derivative Controller
ROS	Robotic Operating System
RSU	Road Sign Unit

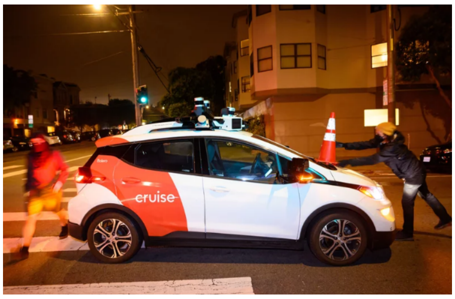
SAE	Society of Automotive Engineers
SAW	Simple Additive Weighting
SDV	Software-Defined-Vehicles
SiL	Software-in-the-Loop
SOC	Security Operations Center
SUT	System-Under-Test
VSOC	Vehicle Security Operation Center
TARA	Threat Analysis and Risk Assessment
V2V	Vehicle-to-Vehicle
V2X	Vehicle-to-Everything

1 Introduction

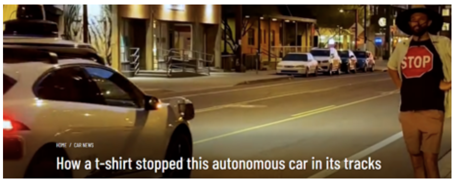
In recent years, the automotive domain has seen a transformation of vehicular architectures from legacy, analog, serial, mechanical control to connected vehicles with software and network-centric architectures. Autonomous Driving (AD) furthers this transformation, replacing human control with software control, guided by intelligent algorithms. AD software relies on input from telemetry of diverse sensors (LIDAR, Camera, Radar, IMU etc.) to create perception of the driving environment and localisation for navigational planning and motion-control [223]. Integrity and availability of sensing input are critical to ensuring the robustness of autonomous control decisions [223] [276] [10]. Security of automotive technologies has been a burgeoning area of research for the last decade since proof-of-concept attacks on controller-area-networks (CAN) and infotainment systems introduced to the public consciousness, the potentiality for an attacker to exploit insecure wireless networks and software vulnerabilities to cause unsafe and adverse driving actions [196]. Within these attack models, the CAN messages for vehicle actuation communicated between embedded electrical (E/E) components can be manipulated to alter the behaviour of the vehicle to produce unsafe outcomes. The attack surface is increased for AD architectures for which the software replaces the human-in-the-loop, and thus the software acts as the interpreter of sensing data, the manager of decision-control and the observer of driving behaviour. Within this software-centric transformation, methods and tools to test the robustness of AD software to cyber attacks and to assess vulnerabilities of decision-control functions is of vital importance in the digital transformation of automotive.

1.1 Problem Statement

AVs have been introduced into real-world driving environments through diverse trials of ride-hailing services [279] [214] [343] and last-mile public transportation shuttles [122] [73] [17]. During the course of their operation these vehicles have experienced a number of adver-sarial events. Activists in San Francisco have used adversarial examples in the form of traffic cones placed at incautious areas of the driving environment (Figure 1a) and stop signs printed on t-shirts (Figure 1b). The aim of these attacks is to induce the object detection to misclassify the adversarial example as an integrous part of the driving environment, interrupt the driving mission, and cause decision uncertainty, effectively immobilising the vehicle [203] [221] [147]. Adversaries are motivated by the challenge of manipulating the autonomous cognition which is supplanting human control and the possibility to induce the algorithmic control to perform actions which impact safety, security and result in an unsuccessful mission.



(a) Safe Street Activist Group Place Cone of Cruise AV [147]



(b) Activist with Adversarial Example T-Shirt [221]

Figure 1: Attacks on AV ride-hailing systems

AVs operating in the road environment are further susceptible to a range attacks. These include usage of lasers to occlude perception sensors (Camera, LiDAR) [302] [159] [5] [88], projection of adversarial examples on the road environment including transparent images [208] and physical invariants (malicious road patches etc.) [316] [114] [178] and jamming and spoofing of GNSS, to name a few. Jamming and spoofing of GNSS is a recurrent feature of driving environments located within and in proximity to areas of geopolitical tensions (See Figure 2).

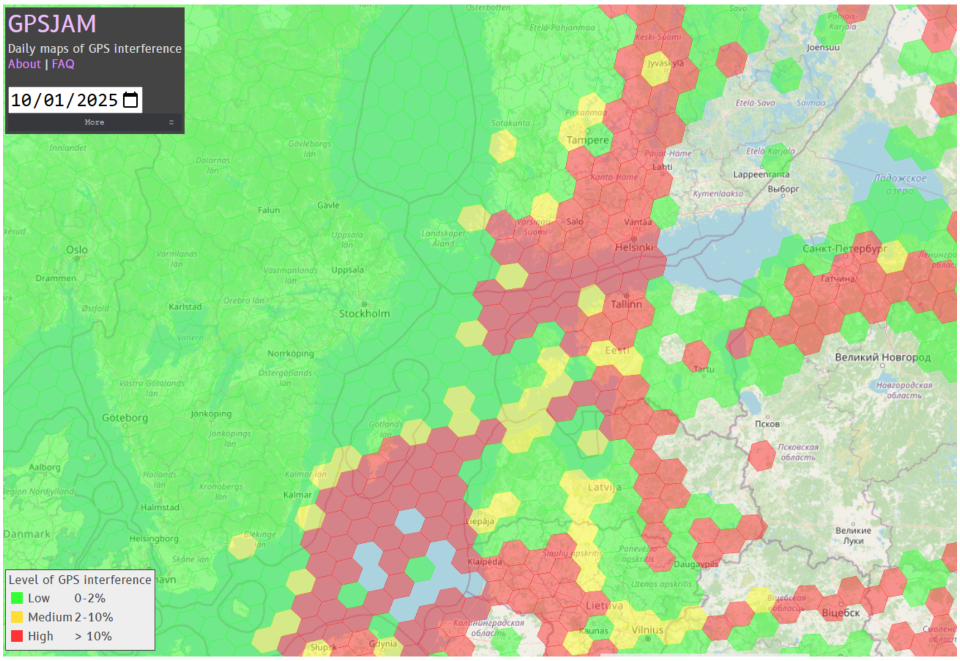


Figure 2: GNSS Jamming Activity within the Baltic States 10th January 2025 [310].

Given the nascent nature of AD technology, there is a predominant need to investigate the vulnerabilities of real-world, operational AD systems to cyber attack, understand how the existing design lacks robustness to cyber attacks and develop mechanisms for testing and assurance for operational readiness.

1.2 Problem Motivation

There is a preponderance of challenges of cybersecurity of AD software. This thesis focuses on three main areas, vulnerability testing, analysis of system impact from cyber attacks and testing methods, tools and processes. The primary motivation for characterising the problem within these areas and orienting the focus of the thesis in this direction is our overall aim which is to assess the vulnerability of a real-world vehicle to cyber attacks and develop knowledge and tangible artifacts which can enhance cybersecurity testing. As evidenced through the aforementioned examples of attacks on real-world vehicles the robustness of the design of AD software is challenged by cyber attacks, particularly, innovative attacks which develop from edge and corner cases. Development of AD technology has predominantly focused on complex areas of system integration and safety validation. Initial supporting technologies such as the ROS middleware [2] and Autoware software framework [4] were designed without mechanisms for secure communication, authentication and integrity checking. Further, within the cybersecurity domain, research is predominantly conducted in a silo, isolated from the knowledge of AD software developers, control system designers and safety validation engineers. There are a sparsity of studies that contend with AD software developed and customised for a real-world vehicle with algorithms optimised for its body physical properties, driving maneuvers and ODD.

Cybersecurity testing of AD is challenged by a lack of comprehensive testing methods which delineate the affect cyber attacks have on system behaviour. Binary pass/fail metrics are the predominant means for evaluating the success of attacks in contrast to metrics which extrapolate a more meaningful evaluation of system behaviour in the context of safety and security [263]. Algorithm designers need to receive detailed feedback from cybersecurity testing to aid root-cause analysis of safety failures. Whether the failure can be attributed to a lack of optimisation of the algorithm to a given scenario-based test or if there exists a lack of robustness to cyber attack. Further, there is limited knowledge on how attacks propagate within this system-of-systems architecture, what crucial breakpoints exist which affect control behaviours and what response mechanisms are available.

At a practical level, to test software used in real-world, operational vehicles, a greater understanding of testbed technologies and tools for structured testing is required. Attack models overwhelmingly are targeted at the physical layer. Such examples include shining lasers into LiDAR sensors, mirror reflections aimed at the camera and adversarial examples targeting road markings and traffic signs. These attacks, are primarily conducted in simulation environments, using generic, off-the-shelf algorithms and vehicle sensor configurations. For applicability to real-world programs, there is a need to investigate the use of diverse testbed technologies including digital-twins which have fidelity to the software stack and sensor configurations of the vehicle. Attacks are constructed as proof-of-concepts, which for usage in operational settings, require customisation and/or reverse engineering to deploy the attack outside its originating environment [263] [223]. Development of structured attack test generation in simulation environments is essential to enable agile and repeatable testing, lower the cost of testing and enable reproducability and wider community usage.

1.3 Thesis Contributions

This thesis provides foundational knowledge for cybersecurity testing of AD software in the context of real-world, operational systems. This work investigates the robustness of AD software to cyber attacks and focuses on addressing a range of key areas of concern. We propose solutions to cybersecurity testing to address the aforementioned challenges that provide greater depth of insights into the robustness of AD software. More specifically, the thesis provides the following contributions to cybersecurity testing:

- We propose diverse iterative and agile AD cybersecurity testing methods. We apply them to a case study of a real-world, operational, AV shuttle.
- We demonstrate the utility of these methods using a testing tool-chain approach consisting of diverse test-bed technologies.
- We discover vulnerabilities in both modular and end-to-end software architectures. Specifically in the planning and localisation software modules of the modular architecture, and the training of neural networks in the end-to-end architecture.
- We evaluate the findings with AD software control designers and safety validation engineers and use this knowledge to understand system optimisation and develop methods for root-cause analysis.

- We present tools for AD cybersecurity testing to enhance structured testing, and community efforts towards standardisation of testing.

The itemised list of open-source tools are as follows:

- ADSecData Platform - Datasets of experiments conducted in this thesis: `https://sites.google.com/view/adsecdataplatform/home`
- REACT - Dynamic intrusion response system for automotive: `https://github.com/AndrewRobertsEst/REACT`
- Self-driving testbed for cybersecurity demonstration videos `https://tinyurl.com/2xxvvkzd`
- ADSecLang - Domain specific language for AD cybersecurity testing: `https://github.com/AndrewRobertsEst/AttackLa`
- FuzzSense - Fuzzing tool for AD software: `https://anonymous.4open.science/r/FuzzSense-E680/README.mdFuzzSense`

1.4 Thesis Structure

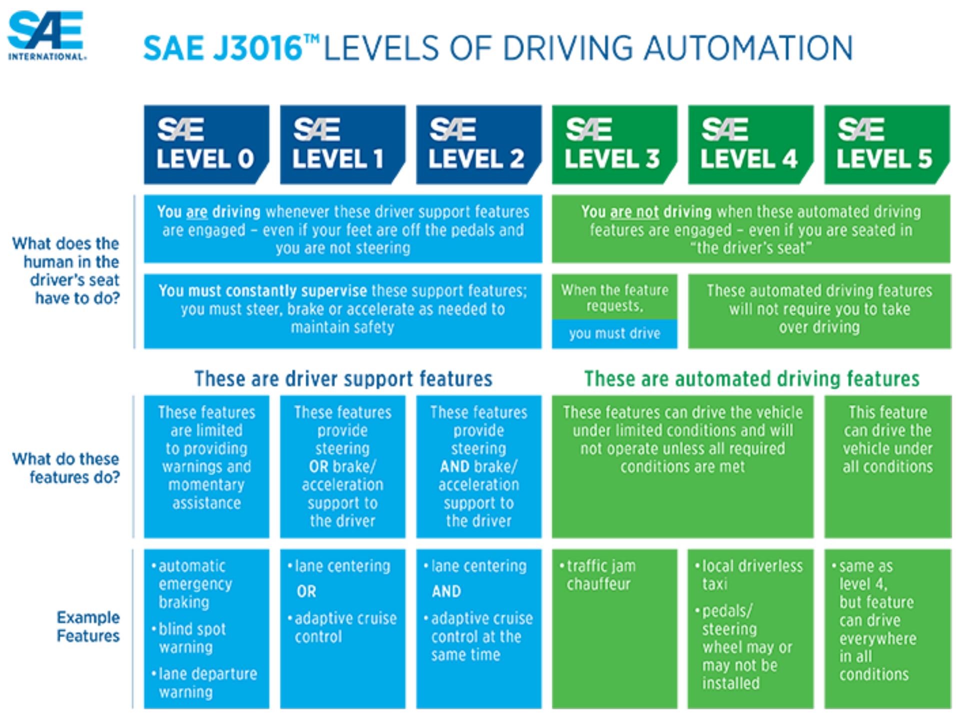
The thesis is divided into 7 Chapters. Chapter 2 details background information about AD software, in particular the foundations of AD software frameworks, cyber threats and a discussion of some related work (Pub. VIII). Next, in Chapter 3, we approach the problem of evaluating affects of cyber attacks within the AD software. In this chapter, we present a methodology for combined security and safety testing and utilise a testing-tool chain approach to explore the problem amidst a range of diverse cyber attacks at the algorithm and sensor level (Pub. IX, VII, IV). Chapter 4 deepens the analysis of cyber attacks in AD software by presenting methods for fingerprinting cyber activity, debugging AD software and investigates mechanisms cyber incident response (Pub. VI, V). Chapter 5 presents conceptual frameworks for structured and fuzz testing and open-data sharing(Pub. X, III, I, II). Chapter 6 provides direction for future work. Chapter 7 concludes with a summary of the findings and their relevance to the broader community.

2 Preliminaries

The architecture of autonomous vehicles consist of a diverse technology stack incorporating from the low-level sensing, electromechanical devices to the high-level software control. To understand the cybersecurity implications of driving autonomy, it is necessary to first understand the technology stack of AVs. Within this section we will present first, a high-level overview of autonomous driving. Consequently, a more detailed extrapolation of the AV technology stack will be presented within the context of the case study AV utilised in this thesis. Last, the cyber threats to autonomous driving are presented.

2.1 Autonomous Driving

To ensure a consistent taxonomy of driving autonomy, SAE [246] have defined diverse levels of autonomy according to technology features and driving control. There are 6 levels of driving automation according to the SAE Levels of Driving Automation (See Figure 3). Level 0 to 2 vehicular architectures are based on human driver control and supervision of the vehicle. Level 0 represents legacy vehicles, there is no autonomy features and software functionality is limited to the provision of warnings and driver assist notices. Level 1 to 2, which is typical of modern connected vehicles, provide limited autonomy technologies such as cruise control, lane-centering and steering, brake and acceleration support. Level 2 is a designation of popular limited autonomy technologies, OpenPilot [22] and Tesla Autopilot [332] (considered between level 2 and level 3). In level 2, the vehicular sensors monitor the driving environment and from the sensing data, lane centering and adaptive cruise control algorithms inform motion control and actuation decisions to maintain vehicle position in the driving environment. The autonomy software does not perform complex driving maneuvers such as overtaking and intersection management and is not able to proactively respond to dynamic driving situations. The human driver is required to keep control and supervision of the vehicle at all times. Level 3 offers limited self-driving functionality with the requirement of human-control when the autonomy functionality encounters uncertainties that it cannot resolve. Full self-driving autonomy is defined within level 4 and level 5. Within these autonomy levels, the autonomous software is expected to control the vehicle without human intervention. This thesis focuses on self-driving autonomy as defined from levels 4 to 5.


SAE J3016™ LEVELS OF DRIVING AUTOMATION

	SAE LEVEL 0	SAE LEVEL 1	SAE LEVEL 2	SAE LEVEL 3	SAE LEVEL 4	SAE LEVEL 5
What does the human in the driver's seat have to do?	**You are driving** whenever these driver support features are engaged – even if your feet are off the pedals and you are not steering			**You are not driving** when these automated driving features are engaged – even if you are seated in "the driver's seat"		
	You must constantly supervise these support features; you must steer, brake or accelerate as needed to maintain safety			When the feature requests, you must drive	These automated driving features will not require you to take over driving	
	These are driver support features			These are automated driving features		
What do these features do?	These features are limited to providing warnings and momentary assistance	These features provide steering **OR** brake/ acceleration support to the driver	These features provide steering **AND** brake/ acceleration support to the driver	These features can drive the vehicle under limited conditions and will not operate unless all required conditions are met		This feature can drive the vehicle under all conditions
Example Features	• automatic emergency braking • blind spot warning • lane departure warning	• lane centering **OR** • adaptive cruise control	• lane centering **AND** • adaptive cruise control at the same time	• traffic jam chauffeur	• local driverless taxi • pedals/ steering wheel may or may not be installed	• same as level 4, but feature can drive everywhere in all conditions

Figure 3: SAE J3016 Levels of Driving Automation [246].

2.1.1 Autonomous Driving Software

Highly automated vehicles, SAE Level 4 to 5, exhibit a sensor layout which consists of diverse sensors (Camera, LiDAR, RADAR, IMU, GNSS) and multiplicity of sensors to ensure dense coverage. This is in contrast to semi-autonomous vehicles which predominantly use camera and ultrasonic radar sensors. The number of camera sensors of semi-autonomous vehicles can range from 3 in the case of comma ai to 8 in Tesla vehicle model [300]. In highly automated vehicles, the AD software performs the tasks of perception, planning, localisation and decision-control. Due to the reliance on algorithms to perform the driving actions and the absence of active human supervision and intervention, highly automated vehicles have the most robust sensing profile for perception and localisation. However, due to cost, especially of the LiDAR and high-definition camera sensors, this architecture is predominantly utilised for autonomous public-transit shuttles and specific commercial use-cases such as logistics and freight [265]. Figure 4 displays the layout of sensors on a level 4 AV for public transportation.

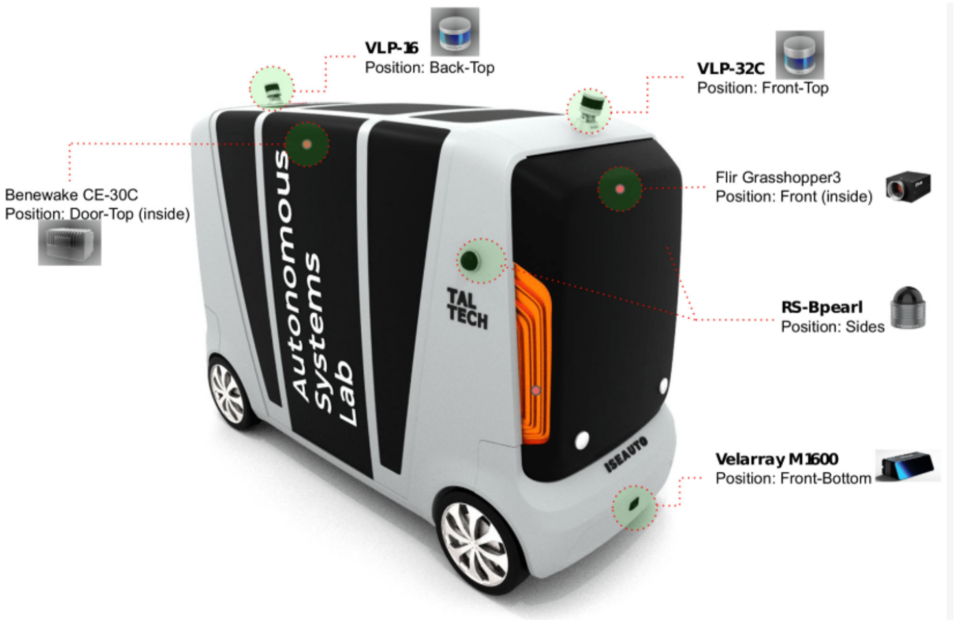


Figure 4: Perception Sensor Layout of Autonomous Vehicle for Public Transport [87].

2.1.2 Autonomous Driving Software Frameworks

Architectures for AD are categorised in three types (Figure 5); *modular/pipeline*, *end-to-end* and *hybrid* [244]. The modular/pipeline architecture compartmentalises the autonomy task pipeline (localisation, perception (detection), prediction, path planning , decision-making, control) into modules. A major benefit of the modular approach is that each of the algorithms for the AD tasks can be accessed and modified according to the requirements of the vehicle development and testing team. This modular approach enables software developers to work on each task, and more clearly understand the inputs, outputs and behaviours of each task in the pipeline. The modular architecture further allows the mixed usage of commercial vendor developed modules with open-source modules. Each module can be seen as the top of a hierarchical structure with sub-modules required for individual processes. A module and its constituents can extend to hundreds of thousands lines of code, considering the complexity of the task. This modularity can also be a disadvantage, if there exists a lack of robustness of one module, it affects that performance of the others. Further, there is additional effort to integrate modular components and ensure seamless communication and performance. The End-to-End architecture uses deep learning to handle the entire navigation pipeline in an unified process. Sensing input is directed into a neural network for processing of navigation decisions. The benefits of end-to-end include a more simplified architecture and more holistic optimisation as optimisation takes place on unified architecture rather than modular parts. Drawbacks include the requirements for training data such as a need for large scale datasets and holistic data that includes edge and corner cases. Also, the lack of transparency of the end-to-end DNN which resembles a black-box. This opacity complicates efforts to debug and troubleshoot. The hybrid approach uses elements of the modular and end-to-end architectures, leveraging neural networks for path-planning, whilst maintaining separate modules for perception and control. The advantages of this approach the targeted use of deep learning for more complex tasks, and deterministic algorithms for tasks that require reliability and interpretability.

The drawbacks of the hybrid approach are complexity of integration and resource usage driven by combined use of deep learning and deterministic algorithms [244].

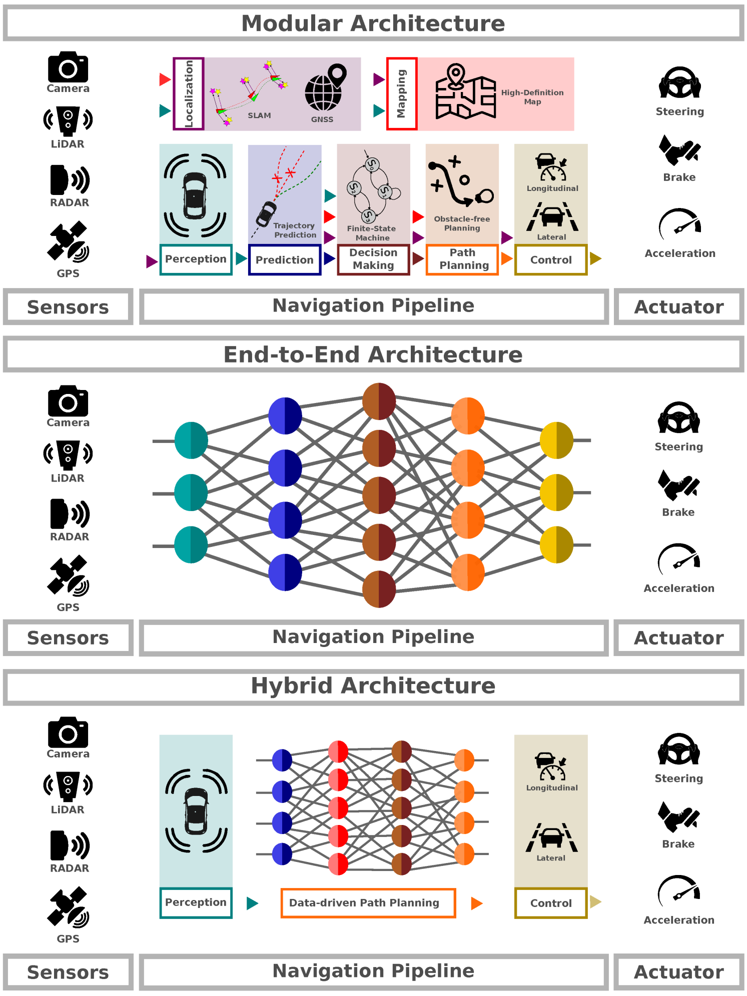


Figure 5: Autonomous Driving Architecture [244]

There are two predominant software frameworks for levels 4 and 5 autonomy, Autoware [4](Figure. 6) and Apollo [15](Figure. 8).

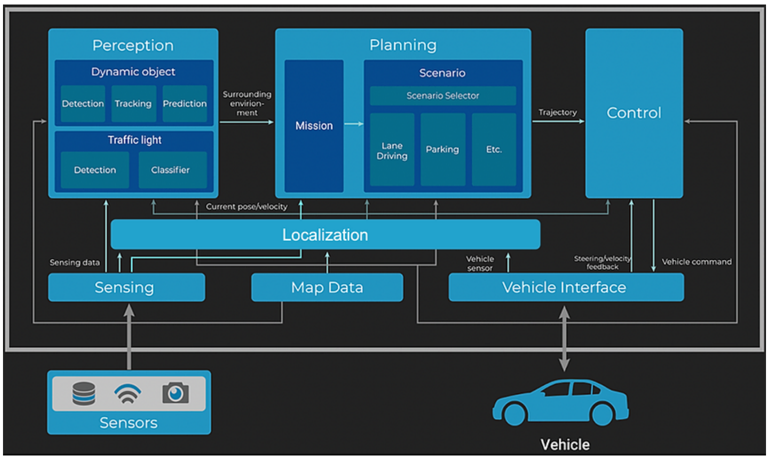


Figure 6: Autoware Architecture [4]

Autoware [4] is an open-source software framework, consisting of a modular/pipeline architecture that encompasses the full-range of AD functionality (localisation, perception (detection), prediction, planning , control) with defined interfaces and APIs. Each of the modules can take diverse sensor input to inform task performance. The development of Autoware is supported by the Autoware foundation which consists of contributors from industry and academia. The AV technology stack of Autoware (Figure 7) is based on ROS [143]. ROS is a middleware that consists of software libraries to support the packaging of hardware and devices and a messaging service to support communication between actuation and high-level software processes.

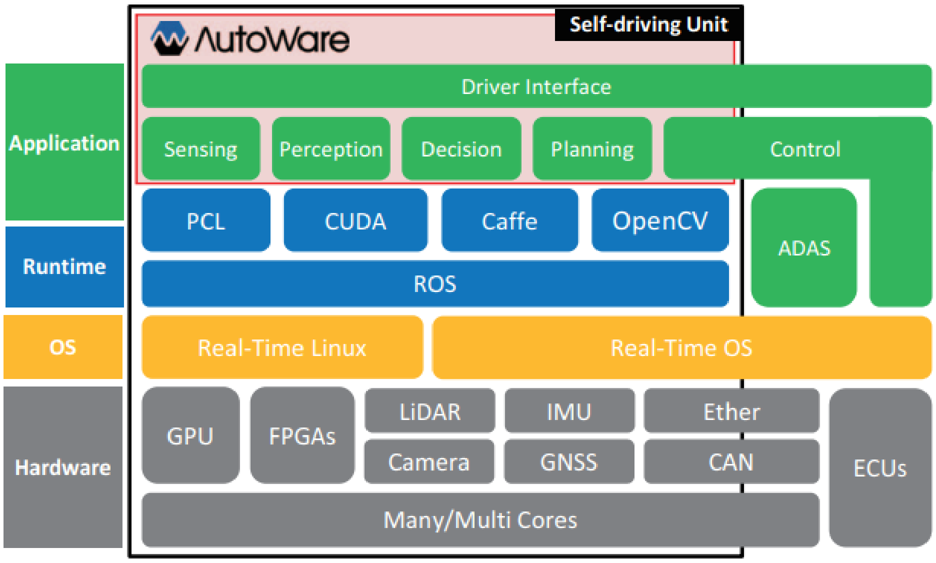


Figure 7: Autonomous Vehicle Technology Stack - Autoware [143]

Apollo [15] is an open platform, end-to-end architecture. The open-platform means that development and testing teams can access the simulation and execute tests, however, the neural network is a black-box and therefore its training and configuration is not open-source. Developer access needs to be granted by Apollo to gain advanced privileges. Cyber RT, like ROS, is the middleware software which underpins the run-time communications required for real-time operation.

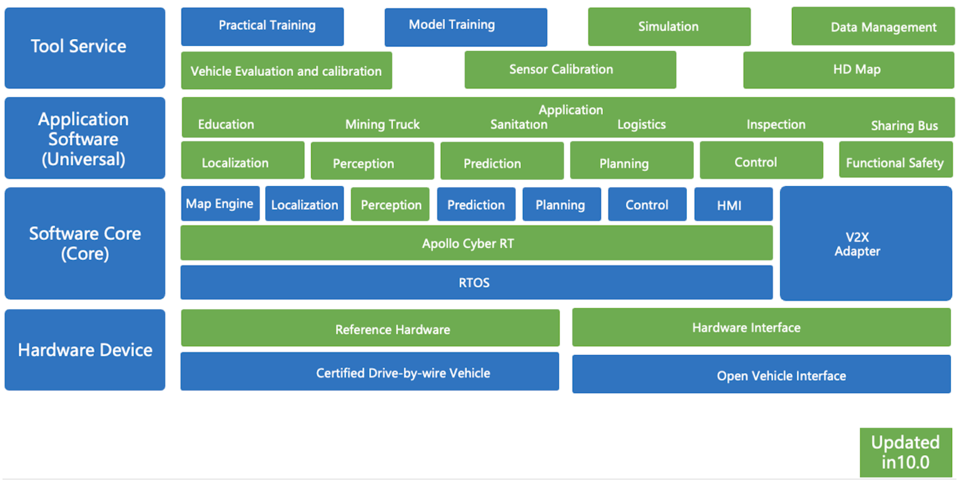


Figure 8: Apollo 10.0 Architecture [15]

2.2 iseAuto: Autonomous Vehicle for Public Transportation

This thesis uses a real-world operational AV system as a case-study. The iseAuto [254](see Figure 9), is a SAE level 4, real-world AV shuttle for public transportation. It provides last-mile shuttle transport for public users in Tallinn, Estonia and has provided similar services for cities in European countries such as Norway and Greece.

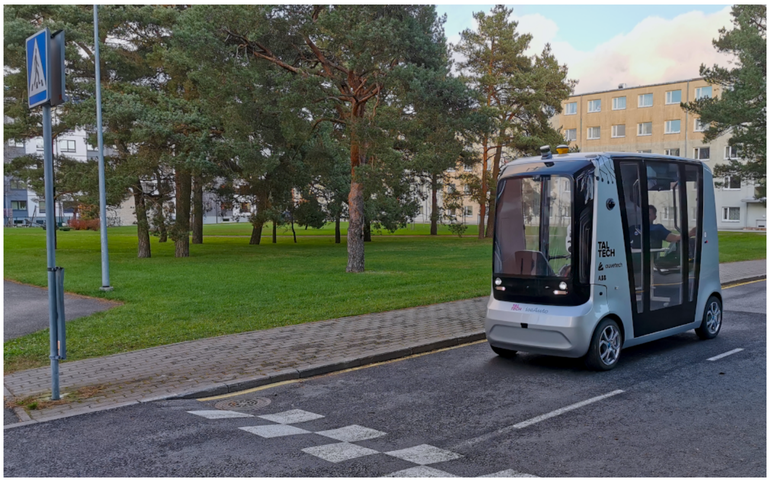


Figure 9: iseAuto autonomous shuttle [185]

2.2.1 Architecture

The architecture of the iseAuto displayed in Figure 10 consists of advanced sensors, the AD platform which contains the AD software, the supporting compute platform and networks, and the low-level control and actuation [256].

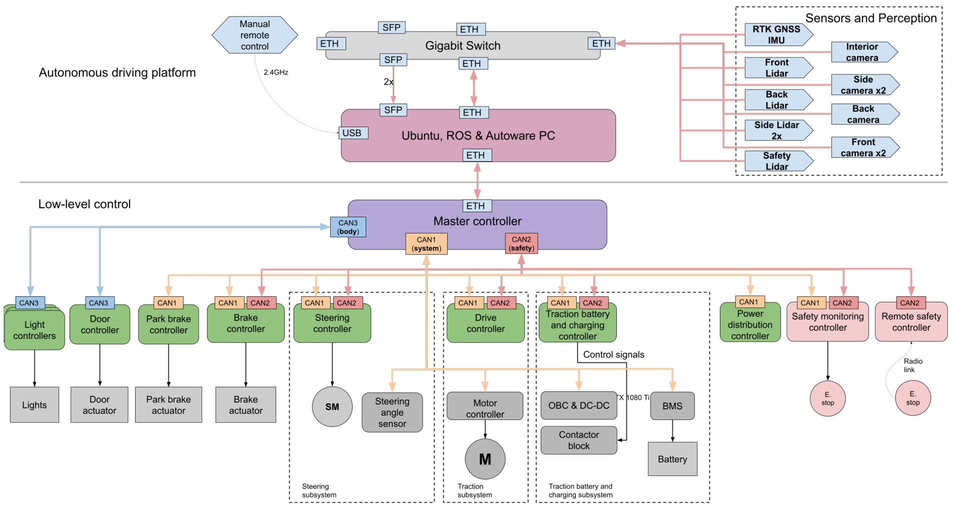


Figure 10: Autonomous vehicle high-level functional architecture [256]

2.2.2 Sensor Configuration

Sensors are connected to the AD platform running AI-based models for identification, detection and segmentation of objects and environmental information through a Gigabit ethernet switch. Data flow is managed and synchronized directly in the Autoware stack, sending data as ROS topics to concurrent threads (nodes) running inference over the AI-based deployed modules. Sensing information are used for perception-related functionalities such as object detection, segmentation and sensor fusion.

The iseAuto uses a multi-LiDAR sensor system for perception and localisation. Two Velodyne LiDARs are mounted at the top front (VLP-32) and the back (VLP-16) of the vehicle, in addition to two Robosense RS-Bpearl at both sides (left and right), to decrease the sensor blind zone around the vehicle. Table 1 lists the iseAuto sensors.

Table 1: Autonomous Vehicle Sensors

Sensor	Model
3D lidar (front)	Velodyne Ultra puck VLP 32
3D lidar (rear)	Velodyne VLP-16
2xSide lidar	Robosense Bpearl
Safety lidar	Ouster OS0-90 (Safety)
3xCamera	Flir
GNSS	Trimble BX992
Radar	TI

2.2.3 iseAuto Autonomous Driving Software

The iseAuto uses Autoware.ai [144] autonomous software stack. As Autoware is a modular architecture, each of the software modules of the iseAuto autonomy platform are customised to the requirements of the vehicle. These include the types of missions, body physical profile and driving environment.

iseAuto uses ROS to communicate with different blocks of the software stack. Figure 11 depicts a ROS-GRAPH displaying only a small subset of the nodes involved in communicating sensing data to the trajectory planning algorithm involved in the overtaking operation.

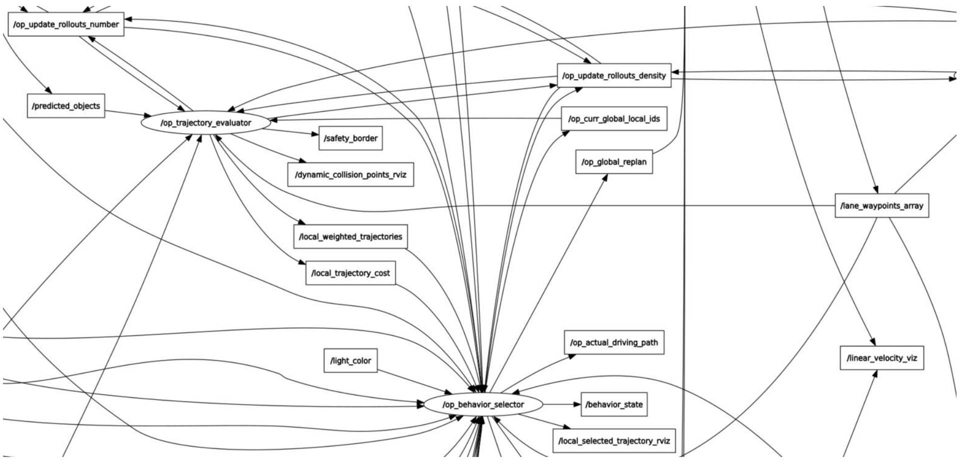


Figure 11: ROS Graph of AV Shuttle During Overtaking Scenario

A part of the ROS nodes/topics running on the vehicle are represented in Figure 12. The software stack is mainly composed of the following main components, sensing and perception, mapping, localization and motion planning. Perception modules runs AI-based modules for detection, segmentation and interpretation of traffic scenes. Localization and mission planning receive constant feedback from vehicle and global positioning to generate new control commands.

Sensing & Perception Module

The algorithm uses the output of the kf_contour_track algorithms to consider all the perceived objects based on the LiDARs point cloud in its local path planning. Earlier, the euclidean clustering algorithm received the filtered point cloud data and prepared point clusters, which is the input of the kf_contour_track. This combination of cluster and contour tracking is done in each sequence for the open-planner to evaluate possible trajectories and create the behaviour based on that.

Localisation Module

This module provides accurate information regarding the position and orientation of the vehicle. Using a *NDT* matching search algorithm, it identifies the best matching position based on sensor perception. It uses input from the *IMU* and the point cloud generated by the LiDAR. Then, it attempts to match the points from our current scan to a grid of probability functions derived from the map. *NDT* matching algorithms can also benefit from the GNSS sensor, which provides initial rough estimates of localization on geo-referenced maps, thereby avoiding any sudden errors in localization calculations that may result in failures. Figure 13 displays the flow of the localisation algorithm within the AD system.

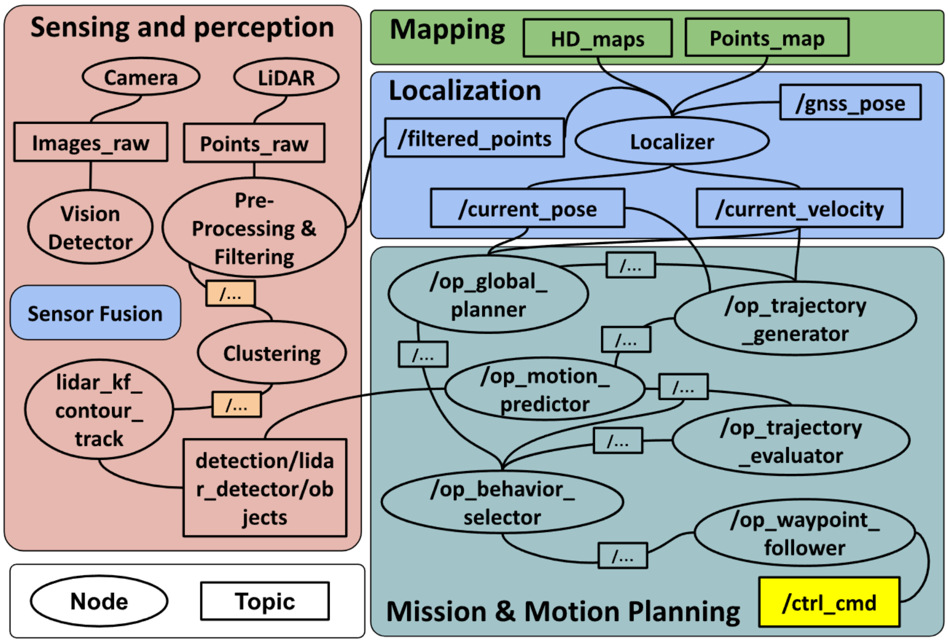


Figure 12: Intelligent driving software stack structure showing ROS nodes/topics communication between essential elements

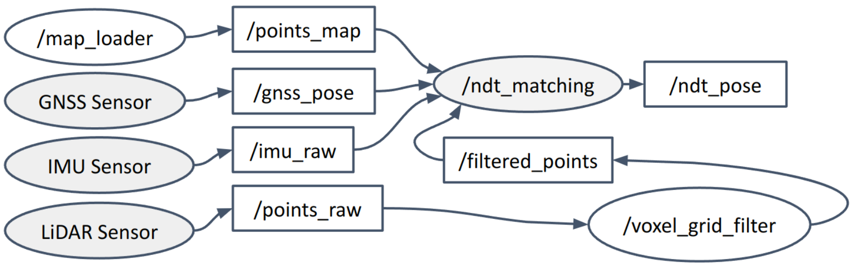


Figure 13: Localisation Algorithm Flow within AD System.

Mission & Motion-Planning Module

The iseAuto uses OpenPlanner [56] as its planning algorithm(see Figure 14). OpenPlanner is one of the most widely used path-planner modules in AD software. In the latest version of this algorithm, which is currently 2.5, the module has become noticeably more advanced in terms of supporting various high-definition map formats, predicting the trajectories of other actors, and using a kinematics-based trajectory generator [56]. This version is compatible with Autoware.ai 1.15. Open-planner combines global and local planners that jointly utilize the road network map to generate local waypoints based on a global route and manage discrete behaviours such as avoiding dynamic obstacles and following traffic lights.

For the AD system to plan a mission, firstly, a global planner generates a global reference path using a vector (road network) map. The function of the global planner is to stipulate a route between the starting position and goal position of the mission on the road map. The local-planner generates smooth and obstacle-free trajectories in the oper-

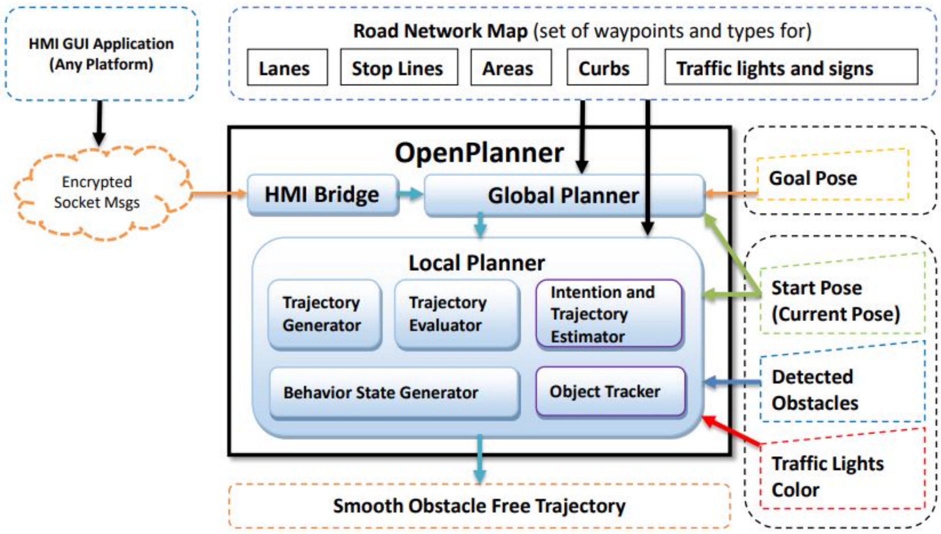


Figure 14: OpenPlanner 2.5 Architecture [57]

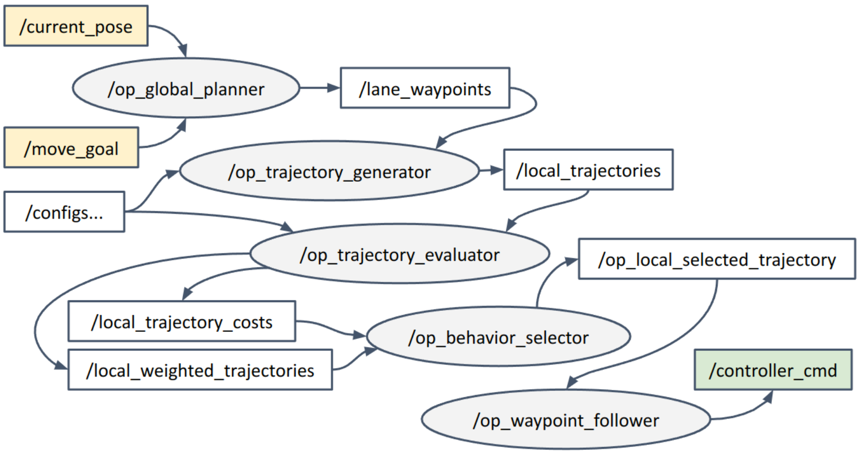


Figure 15: Abstract Local Planning Algorithm Flow within AD System.

ational local domain following the global route.

The local-planner consists of several modules (see Figure 15); trajectory generation, trajectory evaluation, intention and trajectory estimator, object-tracker and behavior selection (decision making) [57].

The trajectory generation module generates alternative tracks parallel to the main path defined by the global planner. These tracks are named roll-outs (see Figure 16). The trajectory evaluation module assesses all possible roll-outs and the data input from sensed-data of the AV and makes a cost estimation. The behaviour selector will lead the AV to motion on a roll-out based on the least-cost. Figure 16 shows how open-planner selected roll-out number 6 in order to pass the non-player character (NPC). It also detects the curb lines and avoids those roll-outs which intersect the curbs.

Table 2 displays the input and outputs of each of the local-planning modules (Note. intention and trajectory estimator and object-tracker are not visible are still developmen-

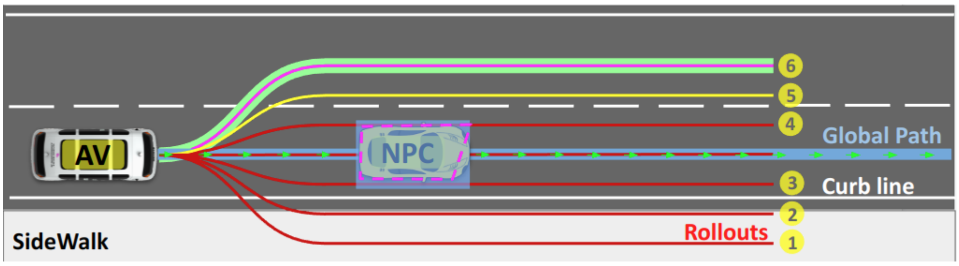


Figure 16: How open-planner generates different trajectory to pass an object

tal).

Table 2: Local-Planning Module

Node	**Input**	**Output**
Trajectory Generator	Initial_Pose	Local Trajectories
	Current_Pose	
	Current_Velocity	
	Lane_Waypoints_Array	
Trajectory Evaluator	Current_Velocity	Local Trajectory_Cost
	Current_Pose	
	Local_Trajectories	
	Lane_Waypoints_Array	
	Predicted_Objects	
	Current_Global_Local_IDS	
Behavioural Selector	Current_Velocity	Current_Behaviour
	Current_Pose	
	Local_Trajectory_Cost	
	Local_Weighted_Trajectories	

Control Algorithm

The local motion planning algorithm generates a trajectory (or a set of control commands for the AV) by minimizing a cost function, within a workspace, that includes a set of design parameters. The cost function constitutes the rules for motion-planning which inform the decision-making for autonomous driving.

The cost function is built on five factors and calculated in Equation 1:

$$C = \begin{bmatrix} w_{cent} \\ w_{trans} \\ w_{longColl} \\ w_{latColl} \\ w_{vis} \end{bmatrix} \cdot \begin{bmatrix} C_{cent} \\ C_{trans} \\ C_{longColl} \\ C_{latColl} \\ C_{vis} \end{bmatrix}^T \tag{1}$$

where, C_{cent} is the cost associated to the central trajectory and is designed to keep the vehicle in the central trajectory; C_{trans} is the transition cost that prevents the vehicle from jumping between roll-outs; $C_{longColl}$ and $C_{latColl}$ are the cost of the longitudinal and lateral collision respectively, and finally C_{vis} is the weight associated to the visibility [207]. Each of these costs are weighted by their respective weighting factors w_i [58].

2.2.4 Low-Level Control

Low-level control is at the base of our software stack having task to give actuators the right commands to generate a desired behaviour. Analog controllers have the function to follow a specific reference signal. It is clear that such signals are measured by transducers and applied to actuators as current or voltage signals to apply a torque to a motor at the low level. The most common and well known analog controller in automotive is the ECU, which regulates injection, speed, and other engine parameters. Brake control modules are also very common and control various aspects of the braking system, such as ABS, ESC, traction control, and brake force distribution. Now, assuming that our goal is to keep any value of cruise speed, a velocity regulator works by receiving a measure of the current speed, comparing it to the reference, and generate a control signal to accelerate or brake accordingly. Low level controllers typically work on a simple control feedback loop involving some type of linear system model (or a linearized one). The most common, state of the art, and well established controller in automotive is the PID controller. They are wide spread in automotive for their simplicity, robustness, usability and real-time capabilities. A PID controller continuously calculate an error signal based on the difference between a desired setpoint and the measured process variable, and then adjust the control output accordingly. They use proportional, integral, and derivative actions to regulate a vehicle actions. The underling equation is relatively easy involving three constants proportional, integral and derivative constants, typically indicated as K_p, K_i, and K_d, to weight the each action respectively. With reference to Figure 10, PID controllers are at the base of the "drive controller" and "steering controller" block. Control theory provides a very stable mathematical theory about analysis and synthesis of the controllers, thus how disturbance might affect the controller is, in principle, well known. This work aim to provide insights on how the behaviour of the controller might affect the decision-making blocks in a real-world, operational AV.

Intermediate Layer/Master Controller

The role of the Master Controller is to parse analog input to the digital network of the vehicle. The Master Controller communicates with the low-level control through the CAN bus. The low-level control section in Figure 10 shows all the basic components in our system, which are connected to the master controller by 3 different CAN busses:

- CAN1 is used to connect all safety critical components, such as brake systems and electric motor.
- CAN2 is used for redundancy over all the safety critical components.
- CAN3 is dedicated to low-priority body-related functions such as door automation and lighting.

The Master Controller receives data from the low level via CAN bus and forwards to the upper-layers via ethernet. Then receives processed signals from the intelligent blocks (the upper-layers) and generate the control commands for the actuators, parsed via CAN bus. Basic data from low-level sensors are processed here and forwarded to the upper layer, this includes speed, acceleration, encoder positioning, voltage and currents. The master controller directly communicates with the upper levels (i.e. AD Software) via ROS topics flowing over the ethernet connected to the Ubuntu-based Autoware PC.

2.3 Cyber Threats to Autonomous Driving

Cyber threats to AD software can be categorised as threats to ***AI semantic components*** and the ***AI system components*** [263]. AI semantic components directly influence the AD model and are defined as the advanced sensor technologies (LiDAR, camera, RADAR, GNSS etc.) which generate input data to the AD model. AI system components are the supporting infrastructure. They are defined as those components which comprise the underlying infrastructure which supports the AD model and the operational state of the AV. Such examples of AI system components include the application software, networks, hardware, and E/E devices. The aim of each threat category is to induce the AD pipeline to influence driving actions which violate safe behaviour.

2.3.1 AI Semantic Components

The adversarial threat models contained in literature, of attacks against AI semantic components, exploit the physical properties of the sensor technology and the semantic properties of the AD algorithms. Advancement in threat research has emanated from foundational work by Eykholt et al. [267] and Petit & Shladover [223]. Eykholt et al. [267] developed adversarial generated robust physical invariants in the form of stickers with pixels manipulated in a manner to that would affect the object detection DNN. These stickers which were placed on environmental objects such as stop signs and pedestrians. They demonstrated that these adversarial examples in the form of physical invariants, could manipulate the logic of the DNN of an object detector to fail to recognise (disappearance) or incorrectly classify (creation) the object. Affects to the vehicle included failing to stop for stop signs and accelerating when the object detector misclassified the adversarial stop sign as an 80 speed limit sign. Petit & Shladover [223] compiled one of the first lists of potential attacks to AD. Many of the innovations in threat research emanated from the directions provided in their paper, especially targeting machine vision and LiDAR. Since the publication of these papers, there has emerged a diverse range of proof-of-concept attacks against the AI semantic components.

Threats to ***camera perception*** *and* ***localisation*** centre on assessing limitations of the camera hardware and the training parameters of the DNN. Threat models include jamming or spoofing light signals using adversary infrared and laser devices. This technique aims to exploit vulnerabilities of the camera hardware and filtering within object detection algorithms such as YOLO and R-CNN [337] [128] [321] [133] [305] [159] [317] [61]. Other attacks include: manipulation of image pixels [267] [47] [101] [212] [206] [161] [27] [46], camouflaging obstacles [335] [114] [302], projecting ghost or transparent images to make them appear physical [209] [191] [178] [5], adversarial generated malicious road patches [319] [316] [132] [47] [134] and manipulating the bounding boxes used for object detection and collision avoidance so that obstacles appear larger or smaller than actual [163] [301] [277] [181]. Attacks against level 2 and level 3 autonomy focus on exploiting the parameters of ADAS and LKAS systems [250] [138]. Studies which generate perturbations of lane-markings have demonstrated vehicles can crash due to low-cost adversarial generated road markings [250] [138] [174]. Table 3 lists threats to the camera sensor.

Table 3: Cyber Threats to camera perception and localisation

Cyber Threats to Camera Sensor			
Paper	**Threat Model**	**Attacker Knowledge**	**Test Environment**
Object Detection			
Lu et al. [128]	Use of different angles and lighting for experiments of perturbed physical invariants.	White-Box	Simulation
Eykholt et al. [267]	Robust physical invariants. Perturbation of Road Sign Units (Stop Sign) using pixelized stickers to confuse object detection.	White-Box	Real-World & Simulation
Chen et al. [46]	Manipulation of image pixels to fool DNN object detection.	White-Box	Simulation
Huang et al. [163]	Manipulation of the parameters of the bounding box to manipulate confidence of R-CNN and YOLO v3 object detection.	White-Box & Black-Box	Simulation
Zhao et al. [337]	Robust physical invariants to manipulate R-CNN and YOLO v3 object detection.	White-Box	Simulation
Xiao et al. [319]	Adversarial generated 3D mesh added to 3D objects to manipulate object detection.	White-Box	Simulation
Zhang et al. [335]	Camouflage vehicles using adversarial physical generation against object detection.	White-Box & Black-Box	Simulation
Nassi et al. [209]	Projected ghost objects on the camera sensor to fool Tesla, autopilot, object detection to perceive them as physical objects.	Black-Box	Real-World
Man et al. [191]	Projection of ghost objects on the camera sensor to manipulate object detection (Yolo v3 and R-CNN).	White-Box	Real-World & Simulation
Wu et al. [316]	Adversarial generated malicious patches to downgrade object detection (COCO and Yolo v2).	White-Box	Simulation

Xu et al. [321]	Adversarial T-Shirts (moving physical invariant) against Yolo-v2 and R-CNN object detection.	White-Box	Simulation
Hu et al. [114]	Adversarial generated camouflage attack against object detection.	White-Box & Black-Box	Simulation
Hamdi et al. [101]	Semantic manipulation of the learning parameters to enable pixel manipulation of the object detection DNN.	White-Box & Black-Box	Simulation
Ji et al. [133]	Manipulation of acoustic signals used for communication with the IMU, triggers motion compensation and blurred camera image impacting object-detection (YOLO v3/4/5, R-CNN and Apollo).	White-Box & Black-Box	Simulation & Real-World
Lovisotto et al. [178]	Projection of physical invariants to manipulate object-detection (YOLOv3)	Black-Box	Simulation & Real-World
Wang et al. [302]	Infrared light used to perturb camera sensor and manipulate object-detection of Tesla, autopilot.	Black-Box	Real-World & Simulation
Köhler et al. [159]	Laser perturbation of camera sensor to distort object detection.	Black-Box	Simulation
Wang et al. [305]	Compresses dimensions of detection boxes to manipulate object-detection (YOLOv3, R-CNN)	White-Box & Black-Box	Simulation
Zolfi et al. [5]	Contactless translucent adversarial generated patch placed against the camera lens to manipulate object detection (YOLOv2,v5, R-CNN).	White-Box	Simulation
Zhu et al. [317]	Placement of lighting bulb on infrared pedestrian detectors to attenuate the lighting to perturb object detection.	White-Box	Simulation & Real-World

Davidson et al. [61]	Spoofing of optical flow sensing of the camera to manipulate the flight of the drone. Development of RANSAC optical flow algorithm enhancement.	White-Box	Simulation
Guo et al. [88]	Projection of modulated light emission from an adversarial source to induce incorrect object-detection of traffic-signs	White-Box	Real-World & Simulation
Ma et al. [182]	Perturbation of video frames of the camera sensor to increase latency of object-tracking	White-Box	Simulation
Semantic Segmentation			
Nakka et al. [206]	Attacker generates perturbations in the image that impact semantic segmentation to cause the object detection to fail to detect road/navigational path from interfering objects.	White-Box	Simulation
Nesti et al. [212]	Adversarial generated road and driving environment (billboard) patches to impact semantic segmentation to cause the object detection to fail to detect road/navigational path from interfering objects.	White-Box	Simulation & Real-World
Object Tracking			
Jha et al. [132]	Manipulation of sensor telemetry through physical attacks (road patches etc.) induce the vehicle to miscalculate distances to pedestrians and other driving obstacles. These manipulations are generated by a DNN to evade defensive mechanisms.	White-Box	Simulation
Jia et al. [134]	Adversarial Examples (Patches) to distract object tracking.	White-Box	Simulation
Ding et al. [65]	Adversarial perturbations of video frames to misguide object trackers.	White-Box	Real-World & Simulation

Chen et al. [47]	Adversarial patch generation to distract object tracking.	White-Box	Simulation
Lane Detection			
Sato et al. [250]	Fool automated lane detected algorithm using an adversarial generated “dirty” road patch.	White-Box	Simulation
Jing et al. [138]	Subtle manipulation of lane markings to fool automated lane detection.	White-Box & Black-Box	Simulation
Traffic Light Detection			
Tang et al. [277]	Tampering with the Region of Interest (ROI) for the automated traffic light detection to fail to detect the traffic light.	White-Box	Simulation
Wang et al. [301]	Adversarial camouflage on driving environment objects to manipulate object detection (Yolo v5)	White-Box & Black-Box	Simulation
Camera Localisation			
Wang et al. [302]	Adversarial infrared sensor perturbation of the camera sensor.	White-Box	Simulation & Real-World

Threats to ***LiDAR perception and localisation*** predominantly focus on injecting malicious LiDAR point clouds into the LiDAR sensor and removing LiDAR point cloud data points. Such attacks are aimed at limitations of the perception and localisation algorithm to filter adversarial sensor telemetry manipulation [35] [273] [286] [341] [325] [340] [171] [284]. Table 4 lists threats to the LiDAR sensor.

Table 4: Cyber Threats to LiDAR perception and localisation

Cyber Threats to LiDAR Sensor			
Paper	**Threat Model**	**Attacker Knowledge**	**Test Environment**
LiDAR Perception			
Cao et al. [35]	Spoofing and manipulation of the input of the LiDAR sensor. Two attack scenarios are implemented, emergency brake attack and AV freezing attack/block traffic.	White-Box	Simulation

Sun et al. [273]	Black-box attacks on LiDAR for general vulnerability testing. Involves inputting randomised adversarial LiDAR points into LiDAR stream to alter perception. Paper develops CARLO detection algorithm.	Black-Box	Simulation
Tu et al. [286]	Adversarial mesh on top of the Autonomous vehicle to obscure the LiDAR detector. Defensive mechanism developed using data augmentation.	White-Box	Simulation
Zhu et al. [341]	Arbitrary objects with reflective surfaces placed around driving location by drones.	White-Box & Black-Box	Simulation & Real-World
Yang et al. [325]	Injection of malicious LiDAR point cloud data through adversarial road-side objects.	White-Box & Black-Box	Simulation
Hau et al. [340]	Manipulation of LiDAR sensor stream through removal of point clouds to disable detection of 3D objects.	White-Box	Simulation
Li et al. [171]	Adversarial spoofing of a AV trajectory with small perturbations.	White-Box	Simulation
Semantic Segmentation			
Tsai et al. [284]	Adversarial generated point cloud data against DNN.	White-Box	Simulation Real-World (Not AV)
LiDAR Localisation			
Luo et al. [180]	Side-Channel attack against cache of LiDAR perception. Attack reveals leakage of data, including location and planning of the AV.	White-Box	Simulation

Threats to ***SONAR and RADAR*** target the transmission of malicious communications on frequencies such as mmWave [275]. These malicious communications take the form of flooding signals and crafted adversarial signals in the specific spectrum band. The aim is to manipulate the SONAR and RADAR to incorrectly interpret a signal as a close object which will feed to the object detection algorithm [266]. Table 5 lists threats to the SONAR and RADAR sensor.

Table 5: Cyber Threats to SONAR & RADAR

Attacks to SONAR & RADAR Sensor			
Paper	**Threat Model**	**Attacker Knowledge**	**Test Environment**
Radar Perception			
Sun et al. [275]	Spoofing of mmWave sensing including adding fake obstacles at arbitrary locations and faking the locations of existing obstacles. Five scenarios are generated in both simulation and real-world environments. Defensive mechanisms are developed using challenge-response authentication scheme and RF fingerprinting.	White-Box & Black-Box	Simulation & Real-World (AV)
Son et al. [266]	Adversarial sound noise to manipulate MEMS gyroscopes.	White-Box	Real-World

Threats to ***IMU and GNSS*** sensors take the form of jamming, spoofing and data manipulation [103] [155] [59] [151] [198]. GNSS spoofing and jamming is prevalent in many operational environments and impacts the reliability of the localisation of the AV which can cause incorrect trajectory planning [59] [198]. Malicious injection of the odometry data (velocity, yaw etc.) of the IMU which includes can impact the trajectory planning of the AV which can have downstream affects to the control algorithm [103] [155]. Manipulation of environmental telemetry (temperature, sensor pressure etc.) can invoke the AV to take safety decisions such as a deploying emergency safety measures [151]. Table 6 lists threats to the IMU and GNSS sensors.

Table 6: Cyber Threats & IMU Sensor

Cyber Threats to GNSS & IMU Sensor			
Paper	**Threat Model**	**Attacker Knowledge**	**Test Environment**
Mit et al. [198]	GNSS spoofing attacks on Tesla Model 3.	Black-Box	Simulation and experiments on real-world vehicle.
Dasgupta et al. [59]	Replication of target vehicle satellite reception to inject stealthy GPS perturbations to alter vehicle course	White-Box	Simulation based on data from real-world vehicle

Han & Zhou [103]	Fuzzing of the GNSS sensor telemetry data input to manipulate the Apollo semantic control program to crash the AV	White-Box and Black-Box	Simulation.
Kim et.al [155]	Fuzzing of the IMU data used for feedback control on robotic systems.	White-Box and Black-Box	Simulation
Kim et.al [151]	Fuzzing of the telemetry of a drone communicating using the MAVLINK protocol. Random generated input is sent to sensor telemetry data inputs for IMU such as barometer, gyroscopy which cause the drone to crash.	White-Box and Black-Box	Simulation

Threat models to sensor fusion algorithms exploit vulnerabilities of the diverse sensor architecture. These include injecting LiDAR point clouds or infrared signals from an adversarial device placed at an angle unobserved by the camera sensor [323] [288] [92] [264]. Table 7 lists threats to sensor fusion.

Table 7: Cyber Threats to Sensor Fusion

Sensor Fusion Perception			
Paper	**Threat Model**	**Attacker Knowledge**	**Test Environment**
Cao et al. [323]	Adversarial 3D printed object that AV fails to detect. This attack is tested against MSF algorithms.	White-Box	Simulation Real-World (non-AV)
Hallyburton et al. [92]	Placement of rogue LiDAR in a specified location near the vehicle, the "frustrum" and injection of malicious point clouds between an angle of invisibility of the camera and LiDAR sensor	White-Box	Simulation
Tu et al. [288]	Adversarial 3D printed object targeting MSF algorithms (LiDAR + Camera).	White-Box	Simulation
Sensor Fusion Localisation			
Shen et al. [264]	GNSS spoofing attack on MSF (LiDAR locator, GNSS Localisation).	White-Box	Simulation

A shortcoming of the AI semantic component threat research include the lack of diversity of target systems. The threats contained in Tables 3-7 use passenger vehicles with a limited sensor profile and off-the-shelf software with no optimisation to the driving scenarios contained in the experiment. Threat models such as Hallyburton et al. [92] exploit a gap in sensor coverage in passenger vehicles which would do not apply to highly-automated vehicles such as AV passenger shuttles.

2.3.2 AI System Components

AI system components include the following:

- Middleware software such as the ROS, CyberRT and others, which enable communication between the hardware and software and facilitate control messages from the higher level AD software to the actuation.
- Networks which enable communication in this densely-interconnected ecosystem and have unique properties, CAN, LIN, MOST, FlexRay.
- E/E components and compute platforms which support ECUs, AD software and other real-time systems.
- The electromechanical components which support the actuation processes of the vehicle.

These components were not designed with security in mind and have been proven to be vulnerable to cyber threats which target the inherent lack of authentication and encryption [232] [153]. Initial threat research on middleware software focused on spoofing the publish-subscribe model of message communication which existed in an environment where there are no mechanisms for a subscribing node to trust a publishing node [232] [153]. This vulnerability of lack of authentication allowed attackers to spoof the master node and issue motion-control commands to low-level actuation, in effect, taking control of the autonomous vehicle [113]. Also, message flooding attacks which create a denial-of-service (DOS) which impact the availability of the message communication transmission which enables safe control of the AV [155]. Other middleware software and communication protocols such as MQTT, which is used for V2X, suffer from the same lack of security in their initial design [94].

Attacks which exploit vulnerabilities of system components aim to manipulate the sensor data input are particularly dangerous as they have the direct ability to affect the AD pipeline. Threats to network communication protocols target the external communication interfaces and the internal communication system. Threat models of the external communication interfaces include manipulating the message exchange between the AV system and the intelligent traffic control infrastructure [77]. Examples include manipulating the geographical location broadcast by the AV system to the intelligent traffic control. This will cause the traffic control to incorrectly estimate the location of the vehicle and make an adverse decision for traffic management [77]. As concluded in Shen et al. [263], there is a lack of cyber threat research on systems unique to the autonomous system architecture and the initial research on ROS have only begun to explore the dependency of the semantic components on system components [208]. There is lack of understanding of the relation of downstream AI system components to the how attacks of the system components (malware, data manipulation and buffer overflows) impact driving decisions [245]. There are numerous surveys of AI system component attacks [201] [63] [85] [327] [62] [172], therefore, Table 8 presents two threats to the AI system components that directly involve experimentation with AD systems.

Table 8: *Cyber Threats to AI System Components*

AI System Components			
Paper	**Threat Model**	**Attacker Knowledge**	**Test Environment**
Hong et.al [113]	Exploitation of ROS publish/subscribe privileges to manipulate sensor data to relocate NPCs into path of vehicle.	White-Box	Simulation
Feng et al. [77]	Manipulation of the geo-location protocol of the V2X to change the location of vehicles and manipulation the traffic management.	White-Box	Simulation

2.4 Autonomous Driving Cybersecurity Testing

The majority of the cybersecurity testing on autonomous transportation systems utilise vulnerability testing methods (See Tables 3-8). Testing can be categorised as white-box, black-box and gray-box. A white-box test is where the attacker has knowledge of the system and is able to use that knowledge to develop an assumption or hypothesis on the vulnerability of the system. The attacker is then able develop a threat model based on this knowledge. A black-box test is where the attacker has no knowledge of the internal processes or architecture of the target system. Gray-box testing is a combination of white-box and black-box, select parts of the target system are known whilst others are opaque. As seen in Tables 3-8, white-box testing is more prevalent. As white-box testing is guided by knowledge of the system, interpretability of results is less challenging than black-box testing where no system knowledge is assumed.

2.4.1 Autonomous Driving Test Platforms

Testing of AD systems are performed on *simulation (SiL)*, *cyber-physical (HiL)* and *real-world testing platforms*. Simulation platforms consists of a rendered 3D virtual environment (which can be customised to replicate the real-world physical environment through 3D LiDAR mapping) consisting of the ODD and AV. The AD software in the simulator, is a digital-twin, which replicates the technology stack of the AV software (Autoware, Apollo etc.). However, limitations of the SiL are that the algorithms underpinning the software need to be customised to the body-physical profile of the vehicle (light-vehicle, shuttle etc.), driving maneuvers and ODD (weather, pedestrians, other vehicles) and the amount of fidelity of the physical properties of the sensors in the simulation environment is an active topic of research [162] [251]. Without access to the AD software of a real-world, validated AV, it is questionable whether a cyber attack conducted in a SiL succeeded due to a vulnerability or a lack of optimisation of the AD algorithms for the type of vehicle, driving maneuver and ODD. Whilst benchmarking/golden run tests are conducted in cybersecurity studies [298], they lack the robustness of safety and software reliability testing due to the extensive amount of tests required and the computational resources involved in generating high-fidelity 3D test scenarios. Therefore, a number of studies use openpilot [22] and Apollo [15]. Openpilot is a level 2 system and therefore not applicable to full-autonomy studies and as aforementioned, the black-box nature of the end-to-end model and the lack of developer access for Apollo complicates debugging failures.

Cyber-physical testbeds (HiL) are utilised where there is a need to observe integration between hardware and software. This is most common in testing the integration between actuation and E/E components with software control. Whilst HiL are commonplace in legacy and connected automotive, there are few examples of AD HiL testbeds. Whilst new contributions such as RAMN [282] exist, where ECUs are fused with AD software, they do not explore integration with more advanced AD sensors (LiDAR, camera) and components (AI computers (Nvidia Drive etc.)).

Real-World testbeds take the form of proving grounds and test track environments. Proving grounds are predominantly used for type approval and test track environments for functional testing and edge and corner cases. These testbeds are the most expensive due to the costs and labor in building and maintenance. Recent research has shown an increase in the use of real-world testbeds to conduct cybersecurity research [264] [224]. This is due to the aforementioned need to improve the understanding of the fidelity of the simulation environments to real-world environments. Safety validation testing in providing grounds has also demonstrated the capability to stream the data from the real-world test to the simulation platform. Thereby, enhancing the fidelity of the simulation. Table 9 lists the simulation platforms for AD.

Table 9: Autonomous Driving Software Simulation Platforms

Autonomous Driving Software Simulation Platforms		
Simulator	**Characteristics**	**Cybersecurity Testing**
AWSIM [1]	+ Open-Source simulator for autonomous driving. Aligned to Autoware Foundation. + Integration with ROS. + Enhanced fidelity to physical properties of LiDAR & camera sensing + Allows custom configuration of driving environment, AV, Sensors. + Scenario Test library integration	+ Sensor Attacks (LiDAR, camera, RADAR etc.). + System Component Attacks (Middleware, software, network etc.)
CARLA [68]	+ Open-Source simulator for autonomous driving. + Integration with ROS. + Allows custom configuration of driving environment, AV, Sensors. + Scenario Test library integration	+ Sensor Attacks (LiDAR, Camera, RADAR etc.). + System Component Attacks (Middleware, software, network etc.)
LGSVL [242]	+ Open-Source simulator for autonomous driving + ROS Integration. + Allows custom configuration of driving environment, AV, Sensors. + End-of-life/Sunsetted + Scenario Test library integration	+ Sensor Attacks (LiDAR, Camera, RADAR etc.). + System Component Attacks (Middleware, software, network etc.)

Apollo [146]	+ Open-Source but for supported for commercial activity. + Apollo 10.0 with Apollo Cyber RT + Custom Scenario Test Library	+ Sensor (LiDAR, Camera, RADAR etc.). System Component Attacks (Middleware, software, network etc.)
GAZEBO	+ Open-Source simulator based on ROS + Limited customisation of AV driving environment, AV, Sensors + Limited Scenario Test Case Integration	+ System Component Attacks (Middleware, software, network etc.)
Air Sim [259]	+ Developed by Microsoft for Drone and Autonomous Vehicle Software Development + Supports diverse autonomy software control architectures (ROS integration, ArudPilot, HiTL, SiTL etc.) + Allows custom configuration of driving environment and sensors. + No test libraries	System Component Attacks (Middleware, software, network etc.
SIM4CV [204]	+ Open-Source, developed for Computer Vision research + Custom configuration of semantic control program + End of life/Sunsetted	+ No Testing has occurred on SIM4CV

The literature of cyber threats to AD demonstrated that approximately 90% of the experimentation was conducted in simulation environments [223] [276] [10]. Yet Eykolt et al. [267] noted that the success rates of attacks in simulation, such as those on the object-detection, differed from real-world. This is most prominent in the physical attacks were the simulation is challenged in replicating lighting and other physical effects. There is a lack of experimentation of cybersecurity testing on real-world systems due to limited facilities and safety risk constraints.

There are numerous AD cybersecurity testing platforms [329] [79] [338] [200]. Whilst these platforms are useful for advancing the research and development of threat models they have sparse usage for validation testing of AD software to cyber threats. A reason for this can be that the design of these testbeds are constrained by a lack of alignment with safety validation testing methods. Further, AD software developers and safety engineers prefer modular tools which can be utilised in their own customised digital-twin simulation environments. Another shortcoming of the cybersecurity testing platforms are that they are technology centric and attack plug-ins are developed for a specific technology stack. There is a lack of overarching principles and methods to guide cybersecurity testing that would enable a standardised approach. Table 10 lists AD cybersecurity testing platforms.

Table 10: *Autonomous Driving Cybersecurity Security Testing Platforms*

Autonomous Driving Cybersecurity Security Testing Platforms		
Simulator	**Characteristics**	**Cybersecurity Testing**
RAMN [282]	+ Cyber-physical testbed for AVs + Integration with AD software + Replicates features of AUTOSAR and automotive networks (CAN/CAN-FD)	+ Testbed was created to support automotive network development, testing of AD software HiL and penetration testing of syntactic software attacks
SEPAD [329]	+ Cyber-physical testbed for AVs + Limited autonomy based on OpenPilot ADAS + Replicates features of AUTOSAR and automotive ethernet + No further development since release	+ Not tested, but test bed was created to support penetration testing of syntactic software attacks
SIMUTACK [79]	+ Simulation environment based on CARLA SUMA (Scenario Generator), OMNeT++ (V2X). + Autopilot for AD Software + Built-in plugins for Attack Generation + No further development since release	+ Attacks to Sensors (LiDAR, Camera etc.). + V2X attacks. + In-Vehicle network attacks.
PASS [338]	+ Simulation environment based on Apollo Baidu and ROS. + Built-in plugins for Attacks and Defenses + Integrates evaluation metrics for safety + No further development since release	+ Attacks to Sensors (LiDAR, Camera etc.) + Fuzz testing + Has supported Capture-the-Flag style, game-based testing
Simulator for Cooperative and Automated Driving Security [200]	+ CARLA and ROS based. + Integrates VEINS V2X network emulation + SUMO Traffic Scenario Simulation + No further development since release	Network Attacks to v2x
AVL Zalazone Test Track [20]	+ Proving ground used for type approval and R&D	+ Cybersecurity testing of V2X and connected infrastructures
Michigan MCity [194]	+ Real-world testbed + Industry and academia R&D	+ Cybersecurity testing of V2X and connected infrastructures + Testing of AD cyber threats to advanced sensor technologies

TalTech Autonomous Systems Lab [18]	+ Real-world testbed + Industry and academia R&D	+ Cybersecurity testing of V2X and connected infrastructures + Testing of AD cyber threats to advanced sensor technologies

2.4.2 Cybersecurity Testing Methods

Testing is predominantly conducted in high-fidelity digital twin simulation environments and then test sets regressed to real-world proving grounds and test tracks. Cybersecurity testing of AD software can be categorised as structured testing or penetration testing and fuzz testing. There are limited methods and tools tailored for structured testing of AD software. Contemporary approaches centre on methods used for connected and legacy vehicles. These entail conducting the TARA and generating cybersecurity test cases targeted at the SUT [192] [239]. Many of these test cases can be extrapolated from proof-of-concept attacks such as those presented in Tables 3-8. The shortcomings of the available proof-of-concept attacks are that considerable effort is required to reverse engineer the attack model and replicate it for different SUTs. Further, the design of these attack models lack consideration for parameters important for safety testing such as temporal aspects of the scenario (time attack should be triggered, how long attack should be broadcast etc.) and scenario design (ODD, driving configurations). There also exists a lack of guidance and standardisation as to how threats can be translated from functional level descriptions to the technical implementation in the digital-twin simulation environment.

Fuzzing is a popular testing technique used to discover vulnerabilities in a system to randomised and unsanitised data input. Fuzzing can occur at three different layers of AD software; the simulator, the scenario and the sensor. A simulation-based fuzzer manipulates the properties of the simulation, this can include GPU and frame refresh rate and CPU settings. A scenario-based fuzzer manipulates the parameters of the driving scenario, these can range from weather (rain, puddles on lanes, snow etc.), odometry (speed, velocity etc.) to planned navigation of road vehicles and pedestrians. A sensor-based fuzzer manipulates the sensor data (LiDAR, camera etc.) which is used as input to the AD pipeline.

There are diverse approaches for the design of fuzzing tools for AD software:

- Adversarial neural networks for adversarial examples targeted at object detection and to generate adversarial trajectories of other road vehicles and pedestrians [170] [339] [281] [333].
- Mutation-based fuzzers predominantly used for sensor fuzzing. They are designed to send unsanitised sensor data input to the AD pipeline [151] [153] [153] [281]. Test cases which cause crashes are added to a seed pool which is used to mutate further test cases in an iterative manner.

Table 11 lists AD fuzz testing tools.

Table 11: Fuzzing Tools for Autonomous Driving

Fuzzing Tools for Autonomous Driving				
Fuzzer	**Target**	**Method**	**Oracle**	**Feedback**
PGFuzz [151]	*Sensor Layer(Drone Software)* + Odometry (velocity, gyroscope)	Mutation-based	Policy-guided (Physical Limitations)	Crashes, Deviation from expected route, sensor inconsistency
RoboFuzz [153]	*Sensor Layer (Drone)* + IMU (yaw, acceleration, speed) and user control commands (throttle, yaw, pitch, roll)	Mutation-based	Physical Constraints	Crashes, Deviation from expected route, sensor inconsistency
RVFuzzer [155]	*Sensor Layer (Drone)* + IMU (yaw, acceleration, speed) and user control commands (throttle, yaw, pitch, roll) messages to control program.	Mutation-based	Physical constraints	Control parameters and control instability
DeepRoad [333]	*Sensor Layer* + Camera images	Neural Network	Predicted Image and predicted steering angle	Object Detection Performance and crashes and deviations of AD
DeepTest [281]	*Sensor Layer* + Camera images during adverse driving conditions (rain, fog etc.)	Neural Network	Predicted Image and predicted steering angle	Object Detection Performance and crashes and deviations of AD
PlanFuzz [298]	*Scenario Layer* + Objects (boxes, bicycles)	Mutation-based	Planning Invariants	Planning behaviour

DriveFuzz [154]	*Scenario Layer* + Weather, mission (location of SUT vehicle) actor (other vehicles, pedestrians)	Mutation-based	traffic rules and regulations	Control behaviour
AVFuzzer [170]	*Scenario Layer* + Route Map, SUT vehicle, weather and objects.	Neural Network	vehicle state (Collision, infraction, mobility)	Control behaviour
AutoFuzz [339]	*Scenario Layer* + Route Map, SUT vehicle, weather and objects.	Neural Network	Traffic Violations and API Grammar	Control behaviour (collisions, infraction)

2.5 Summary

AVs represent a dense ecosystem of diverse software and hardware technologies integrated by an overarching AD software framework (Section 2.1). The research community has contributed an initial list of proof-of-concept attacks and vulnerabilities of AD software (Section 2.3). The predominant attack targets are the sensing and perception hardware and software modules and the networked infrastructure which supports the algorithmic AD platform. Limitations of this initial research include a lack of analysis as to how cyber attacks propagate through the AD software and affect decision-control and knowledge as to how attacks can be applied to real-world systems rather than open-source simulations. Further, there is a lack of deeper investigation of the testing technologies which support cybersecurity research and validation testing (Section 2.4). These limitations are meaningful as this lack of knowledge leaves questions as to the utility of current methods, tools and testing results to real-world AD programs. Therefore, there is an apparent need to develop methods and tools to enable more repeatable and agile testing and to gain from test results greater intuition as to the robustness and resilience of AD software to cyber attacks. Within this thesis these gaps will be explored within the context of an experimental case study of a real-world, operational AV (Section 2.2).

3 Evaluation of Autonomous Driving Software to Cyber Attacks

3.1 Methodology for Combined Safety and Security of Autonomous Driving Software Testing

Testing AD algorithms for performance under safety test cases is a predominant focus for developers to assess the reliability of the algorithm and for optimisation. AD algorithms are also susceptible to manipulation from cyber threats which target the advanced hardware technologies sensor telemetry which serves as an essential input for perception, detection, and control decisions [27, 174, 325]. Existing methods [35, 92] for testing are challenged by the complexity of evaluating system-of-system interactions to identify key relationships and parameters, and limitations of testing inherent to real-world AV programs, resource usage and time. The main idea of this research is to establish a method for combined safety and cybersecurity testing of developmental AD algorithms to evaluate system-of-system interactions to identify and investigate parameters that impact safety and the effect of cyber attacks, and to develop future ideas for optimisation of testing. To develop such a method, we are interested in three research questions aligned with the challenges of combined safety and cybersecurity for AD algorithms:

RQ1 How can AD algorithm designers evaluate the reliability and optimisation of the AD algorithm to both safety and cybersecurity test cases?

RQ2 How can combined safety and cybersecurity testing be conducted on a developing AD algorithm?

RQ3 What key relations and parameters can we identify that can optimise safety and cybersecurity testing?

To evaluate these research questions, we apply the methodology to a developing AD algorithm in a digital twin, SiL simulator and real-world AV testing environment. Cybersecurity testing and safety testing are often conducted separately, reducing our understanding of the relationship between failures of the algorithm caused under normal safety scenarios and failures caused by the impact of cyber attacks. For AD algorithms in the development stage, where the reliability and optimisation of the AD algorithm to safety scenarios have not been established, this exploration of the relationship between safety and cybersecurity can offer novel insights to improve the awareness of the AD algorithm designer to shortcomings in the algorithm.

3.1.1 Combined Safety and Cybersecurity testing methodology for AD Algorithms

The architecture of the proposed combined testing methodology is presented in Figure 17. This method takes advantage of a high-fidelity SiL simulation [255] approach to validate and verify the performance of a AD software under critical cyber security conditions. This method consists of three main following elements:

- Attack script: which simulates a critical security condition.
- High-fidelity simulator: It is a game engine environment that provides the physics for modeling sensors and motion.
- AD software: It is the autonomous driving software that controls the AV.

The combined safety and cybersecurity methodology consists of the following iterative steps:

- **Scenario Selection:**
    - Selection of driving scenario (intersection, overtaking manoeuvre etc.)
- **Analysis of the scenario to extrapolate the safety evaluation criterion applicable:**
    - Selection of safety evaluation criteria is based on relevance to scenario i.e a straight line navigation will not require distance-to-collision criteria metric as there is no other vehicles.
- **Safety Test Case Setup:**
    - Initialisation of the SiL high-fidelity simulator and configuration to the real-world AV
    - Initial scenario testing using the safety test cases to assess the reliability of the algorithm and the quality of the test data
    - Optimisation of the safety test cases to select a subset of the scenario tests to assess the reliability of the algorithm
    - Run of the safety test case scenarios
    - Selection of distinct safety test case scenarios which provide most stable results in terms of success of mission and safety violation
- **Cybersecurity Test Case Setup:**
    - Analysis of the scenario to determine cyber attack strategy for test cases
    - Development of the code for adversary generation in the SITL high-fidelity simulator
    - Selection of attack parameters
    - Optimised the cybersecurity test cases
    - Evaluate cybersecurity test cases in SiL high-fidelity simulator
    - Real-World AV Testing for safety and cybersecurity
- **Results Analysis:**
    - Analysis of the performance of AD algorithm to safety criteria
    - Analysis of sensitivity of attack parameters and driving parameters

Testing Environment
All tests are conducted in a virtual environment powered by the "Unreal game engine" (Unreal) [40]. CARLA simulator [69] is one of the open-source high-fidelity vehicle simulators capable of connecting to different AD software and scenario generator applications. In this study, we use Carla 0.9.13 as the high-fidelity simulator. Figure 17 illustrates the requirements for the high-fidelity simulator to conduct simulation testing which are two components, the digital twin of our AV and the virtual replication of our target environment. These replicated components help us to gain more accurate results of the proposed platform [187]. The AV digital twin is a 3D model of our real-world world AV shuttle, designed in Blender, a graphical 3D modeling software, and imported and built in Unreal for deployment in CARLA. This model uses the same dimension and sensor configuration (model, position, and orientation) from the real AV shuttle. The environment digital twin, in our case, is identical to the location where we are testing and operating our shuttle,

this includes the urban details and vegetation. The next module in the simulator is a scenario generator that produces the desired scenario based on the user input specification. Finally, the simulator engine generates sensor data from sensors, including LiDARs, cameras and others and publishes it for other blocks (see Figure 17 the simulator block). Then, the AD software receives this data as raw LiDAR point-cloud information and processes the data as mentioned in the diagram (Figure 17).

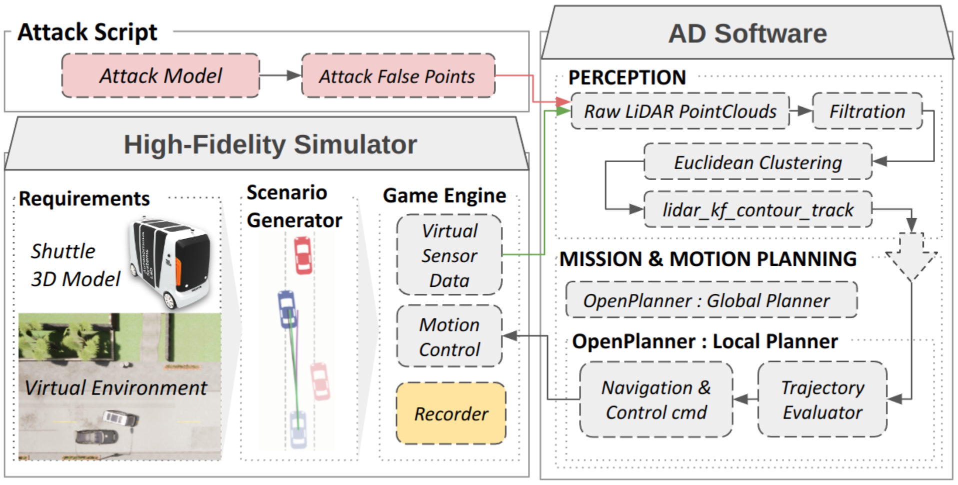


Figure 17: Architecture of the testing platform

This simulation setup was implemented on a desktop computer with the following configuration:

- Intel® Core™ i7-11700K @ 3.60GHz × 16 cores
- NVIDIA GeForce RTX 3080 10 GB
- RAM: 128 GB

Scenario Selection

To evaluate the combined safety and cybersecurity testing, we chose a simple overtaking manoeuvre, which is one of the most safety challenging operations [186]. Figure 18 shows the functional level of the planned scenario. To generate a variety of distinct scenarios, we opt for the initial relative distance to the NPC D_x and the NPC constant speed S_{NPC} as the distinct scenario parameters.

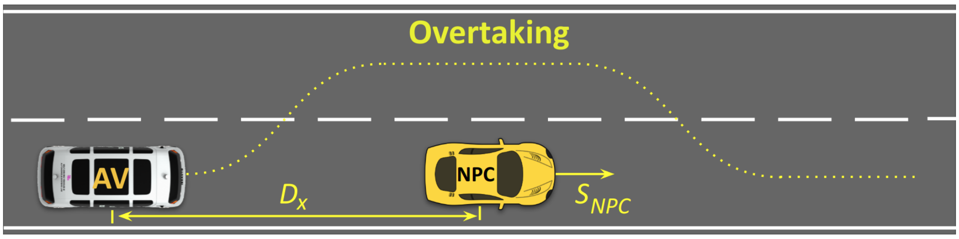


Figure 18: Overtaking Scenario and parameters

Table 12: *Target scenarios definition*

Actor	Speed	D_x	Goal
AV	[0:6]m/s	0 (m)	overtake the NPC safely
NPC	[1 1.4 1.8 2.1 2.5]	[15 20 25](m)	keep moving

Safety Evaluation Criteria

In determining the evaluation criteria for AV safety we considered two conditions, 1) mission success and 2) safety violations. A safety violation consists of a collision and dangerous driving behaviour. In determining which criteria to apply, we considered the EuroNCAP [3] and ISO26262 [127] standards as well those used in composite studies [35, 89, 92]. We derived that the safety goal of the AD algorithm is to execute the overtaking mission without colliding or interfering with other ego vehicles or objects and without exhibiting driving behaviour which is dangerous to the AV passengers. Table 13 details the safety criteria applied in our experiments.

Table 13: *Safety Evaluation Criteria*

Safety Condition	Data Label	Description	Metric
Succeed	Suce	AV Successful complete the mission	Pass/Fail
Not Finished	NotF	Failure to finish the mission	Pass/Fail
Distance-to-Collision	DTC	Violation of the safe distance between AV and NPC	AV within 0.5m of other vehicle
Break on Driving Lane	BrD	AV initiates emergency break on driving lane	Pass/Fail
Break on Passing Lane	BrP	AV initiates emergency break on passing lane	Pass/Fail
Collision	Col	AV collides with NPC	Pass/Fail
Violation	V	Safety Violation	

Safety Test Case Setup

To evaluate the reliability and optimisation of the AD algorithm for the overtaking manoeuvre, we, firstly, initiated a run of 50 distinct scenarios in the high-fidelity simulator, repeating 6 times. Each scenario was repeated 6 times to ensure the reproducibility of the outcome. With the mentioned desktop configuration, it took approximately 100 sec for each scenario and, in total, 8.3 hours for 300 runs. The purpose of the first scenario run was to provide a general overview of the performance of the algorithm. We targeted a range of 1 to 3 m/s for the NPC speed and 15 to 30 m for the initial relative distance to the NPC for selecting the 50 distinct scenario parameters. The results showed that the AD algorithm could not safely overtake the NPC at an NPC speed higher than 2.5 m/s and a distance (D_x) of more than 25 m.

Although a high number of scenario variations shows better coverage in the scenario space to find corner cases, it will lead to an increase in the time duration of the runs. Furthermore, the number of each scenario repetitions was not sufficient to statistically explain the occurrence of each safety violation. Finally, it is worth mentioning that, as our primary study focus is not just the validation of the AV performance, we need to use an optimum number of trials for both safety and cyber test cases. Due to this, we limited the scenario parameters space to the intervals listed in Table 19 that regressed the test set to 15 distinct cases in a full factorial setup. This enabled us to repeat the simulation of these test cases 50 times and apply the full set of safety criteria: collision, DTC, break in passing lane, break in driving lane, failure to finish, and mission success.

Each scenario is generated by the CARLA scenario runner utilizing the Python behaviour trees to handle series and parallel events in the scenario. Figure 19 depicts the scenario scheme starting with the main sequence behaviour. This series begins with transforming the actors into the environment and finishes by destroying the actor block. A parallel behaviour (Driving Toward Intersection) is defined to run the attack and the scenario stop block while the NPC follows the defined waypoint. For safety test case scenarios, the attack block is skipped, and the scenario waits till the stop criteria are satisfied.

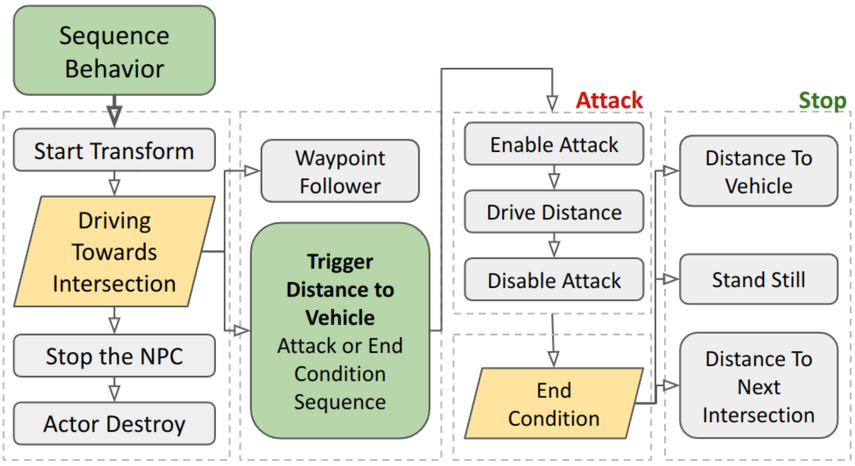


Figure 19: Flow-graph of how each scenario is processed in the simulation platform

Cyber Test Case Setup

To determine the cyber attack strategy for implementation in this test scenario, we analysed the overtaking scenario and its applicability to state-of-the-art attacks on AD algorithms. We selected LiDAR spoofing as it is a realistic attack in the driving environment of our real-world AV shuttle [35] and its impact is relevant to safety outcomes due to the likelihood that the manipulated driving behaviour will result in collisions, emergency breaking, and lane violations [325]. Attacks on LiDAR perception predominantly focus on spoofing LiDAR 3D point-clouds through the following means: 1) injection of adversarial LiDAR 3D point cloud data to add adversarial objects to the driving environment inducing a *false positive result* of the AD perception [35, 273] 2) removal of LiDAR 3D point cloud data to perturb the ability of the perception algorithm to detect objects in the driving environment, also known as a *false negative result* [92, 340] 3) manipulating LiDAR 3D point cloud data to obfuscate the true distance of environmental objects (Other road vehicles, pedestrians, other road objects) from the AV, causing the perception to *fail translation* 4) implementation of adversarial mesh in the driving environment to introduce manipulated points into the LiDAR 3D point cloud and create unpredictable perception events [287]. The aim of the attacker, in adversarial LiDAR threat models, is to induce the victim AV to perform dangerous driving manoeuvres, which include; emergency breaking, collisions, and exceeding the limits of the driving lanes. Variables that have been shown to influence attack success include; angle of attack of the adversarial point cloud vector, density of the spoofed points, duration of the broadcast of spoofed points, distance of the point cloud to the target [35, 92, 273, 325]. We implemented a variation of the attack suggested by Yang et al. [325], where the adversary creates an adversarial roadside object to inject spoofed, malicious LiDAR point clouds into the target AV LiDAR. In our attack, an adversary has configured a LiDAR on the roadside to inject malicious point cloud data into the AV as it is conducting the overtaking manoeuvre. Using the knowledge gained from literature [92, 273, 325], the parameters we chose to generate our attack are: density of the LiDAR point clouds, frequency (the publishing rate of the fake points), duration of the adversarial point cloud broadcast, and location, which is the relative location between the target vehicle and NPC. As an infinite number in the range of each of the parameters can be chosen, we decided to limit our testing to parameter values that had demonstrated utility to investigate the impact of cyberattacks on AD algorithms. For example, Hallyburton et al. [92] found that the success of cyber attacks increased when spoofed point density were over 80. Therefore we chose a range for spoof point density from 50 to 300.

Taguchi Analysis

In this study, we use the Taguchi method for statistical evaluation [285] of the attack parameters effect on each safety criterion. The number of tests with four parameters and 3 levels for each in full factorial mode would become unrealistic to perform, noting that each experiment should repeat 50 times (81x50 = 4050 distinct scenarios). A design of the experiment is recommended in order to avoid full factorial tests and reduce the number of tests without compromising accuracy [285].

A Taguchi design of experiment (DOE) technique [285] was applied to quantify the influence of four proposed attack parameters; the false points (FP) density, the FP frequency, the attack duration, and the attack location. In total, 9 experiments were designed with 3 different values for the four parameters. The analyses hence possess four factors and three levels for the Taguchi L9 matrix. Table 14 lists the configuration for each run conducted for cybersecurity tests.

Table 14: Taguchi L'9 matrix for study of factor influence

Num.	Density	Frequency	Duration	Location
1	50	5	3	3
2	50	7	6	6
3	50	10	9	9
4	150	5	6	9
5	150	7	9	3
6	150	10	3	6
7	300	5	9	6
8	300	7	3	9
9	300	10	6	3
	[50 150 300]	[5 7 10]	[3 6 9]	[3 6 9]

Figure 20 demonstrates the cyber attack setup within the overtaking scenario (Please note, the Figure only depicts the overtaking frame and not the entire overtaking sequence.). The proposed attack model will start by generating spoof points from the designated place on the roadside. At the starting point, P_1, the AV has relative distance to NPC that defines the attack location. After a specific duration (Attack Duration), the AV reaches, P_2. While the attacker keeps the malicious LiDAR pointing toward the AVs front LiDAR. Overall, the spoofed point direction changes from θ_1 to θ_2. Code was created for the generation of the adversarial LiDAR fake points to be run in the digital twin, high-fidelity simulation environment. This is available on the GitHub site [188].

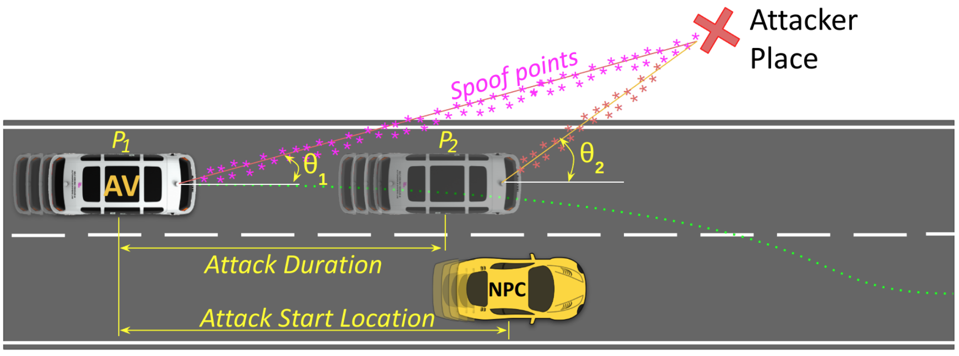


Figure 20: Attack scheme

3.1.2 Results and Analysis

In this section, we present the results of the safety and cybersecurity testing of the AD algorithm. The purpose of the safety test case results is to evaluate the reliability and optimisation of the algorithm.

Safety Test Case

The aim of the testing is to assess the utility of the methodology to evaluate the relationship between the reliability of the AD algorithm to safety and the impact of cybersecurity. As the testing is based on a real-world AV, we were motivated to establish what results could be gained from an amount of tests that took into account the requirements for CPU and GPU resources and the time involved in running high-fidelity simulations. For instance, 50 distinct scenarios run 3 times expends x amount of resources, and takes x amount of time. Therefore, we, firstly, performed a baseline evaluation test where we ran 50 distinct scenarios of the overtaking manoeuvre, 3 times. Each scenario is distinct based on changes to parameters such as NPC speed and initial distance to NPC.

In our proposed simulation platform, we perform 15 distinct scenarios, run 50 times; in total, 750 consecutive simulation runs were conducted. Table 15 shows the parameters of the distinct scenarios evaluated against the safety criteria. Using our configuration for testing, the AD algorithm shows the performance for the overtaking manoeuvre with a success rate of 43.9$\%$ of the simulated scenarios, whilst, 66.1$\%$ are safety violations.

Figure 21 displays the performance of the AD algorithm. NPC speed is an important parameter as it influences the decision control for the critical cut-in manoeuvre of the overtaking mission. In the context of the results of the simulations, we can see that NPC speed impacts certain safety criteria. The first such relation that can be seen, is that more collisions are caused at high speeds, > 2.1 m/s. This can be the effect of a poor trajectory evaluator that doesn't consider the prediction of the other actors motions in the process of the waypoint generation. In most collision cases the AV tried to perform a cut-in while the NPC collided from the right side. The probability of this safety violation will be increased as the NPC speed increases. NPC speed also impacts the likelihood of a DTC safety violation. In the range of the NPC speed parameter, 1 m/s to 1.8 m/s, it can be observed that AV Shuttle violates the safe distance to the NPC. This can be due to the AV speed adjusting relative to the NPC speed and the cut-in is attempted at low-speed, whilst acceleration is required to safely attempt the cut-in. This low-speed cut-in firstly causes a

Table 15: Summary of the safety simulation

	D_x	S_{NPC}	V_{Col}	V_{DTC}	V_{BrP}	V_{BrD}	V_{NotF}	V_{Suce}
1	15	1	18%	22%	0%	10%	24%	26%
2	20	1	18%	40%	8%	6%	18%	10%
3	25	1	4%	20%	32%	8%	20%	16%
4	15	1.4	6%	32%	16%	2%	12%	32%
5	20	1.4	22%	26%	14%	6%	2%	30%
6	25	1.4	4%	12%	22%	8%	0%	54%
7	15	1.8	36%	34%	8%	2%	6%	14%
8	20	1.8	22%	12%	2%	2%	0%	62%
9	25	1.8	18%	6%	0%	4%	0%	72%
10	15	2.1	4%	0%	4%	2%	4%	86%
11	20	2.1	8%	10%	0%	0%	0%	82%
12	25	2.1	24%	0%	0%	4%	0%	72%
13	15	2.5	14%	6%	0%	6%	2%	72%
14	20	2.5	44%	22%	14%	0%	2%	18%
15	25	2.5	64%	18%	0%	0%	6%	12%
		mean	20.4%	17.3%	8.0%	4.0%	6.4%	43.9%
		STD	16.8%	2.3%	9.8%	3.2%	8.1%	28.3%
		min	4%	0%	0%	0%	0%	10%
		max	64%	40%	32%	10%	24%	86%

DTC violation and if the overtaking manoeuvre progresses it causes a collision. DTC and collision correlate based on the relative speed. A low-speed NPC will likely result in a DTC violation, whilst in a higher-speed scenario, a collision is more likely to happen.

In the lowest speed range, 1 m/s to 1.4 m/s, it is more likely that the AV will initiate an emergency break in the passing lane. This is due to the relationship of the NPC speed to the AV Shuttle speed. The emergency break on the passing lane at low speeds is caused by a failure of the open-planner trajectory evaluator to effectively plan the overtaking trajectory. Figure 22 demonstrates the AV emergency break in the passing lane, for a scenario with an NPC Speed of 1 m/s. The upper rectangle represents the AV and the lower rectangle is the NPC. The two rectangles closest to the left represent the frame that the first emergency break on the passing lane safety violation occurs. The most right rectangles represent the end of the mission. The AV speed and the acceleration verify two hard brakes in the mission while it was in the passing lane. The failure of the trajectory planning of the open-planner algorithm is apparent.

The failure to finish the overtaking mission is most prominent at the lowest speed, 1 m/s, this is due to the time the AV Shuttle is taking to perform the cut-in process and therefore cannot enact the overtaking manoeuvre within the simulation timeout which is 40 s. It was observed that for the proposed configuration, for the lower speed of the NPC, the open-planner trajectory evaluator is not reliable as it suggests waypoints that are not within safe navigation and this is due to the lack of firm decision-making of which roll-out to choose. Ultimately, this causes collision and DTC safety violations. Furthermore, the failure to finish the simulation results, we see the low-speed delays in the overtaking manoeuvre decision making which results in the breach of the 40 s time-out.

The success rate of the safety test cases increases as the NPC drives from 1.4 to 2.1 m/s speed. This focal success point around scenario 10 with an NPC speed of 2.1 m/s can be a sign of matching the current configuration of perception and open-planner with the

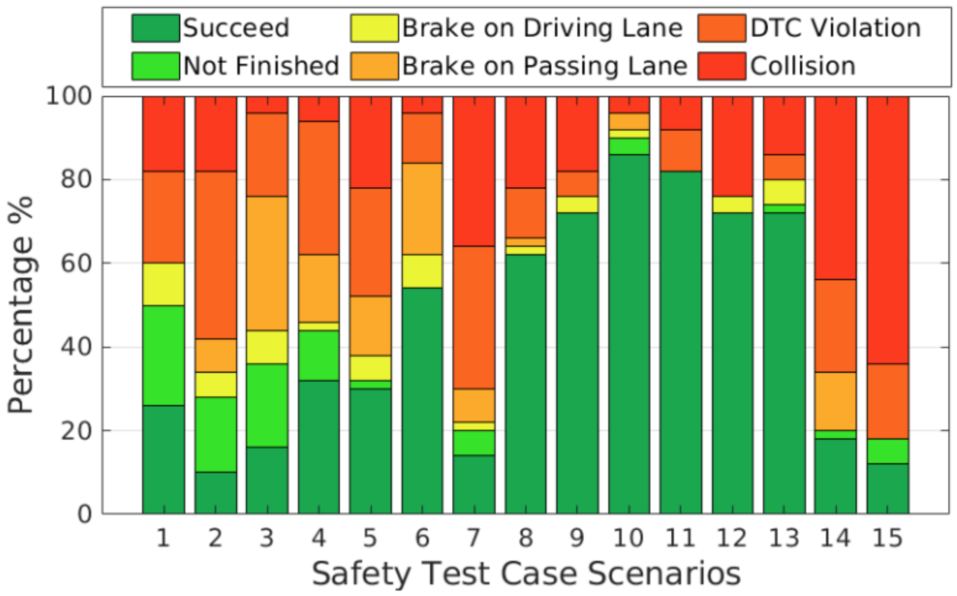


Figure 21: Safety results of 15 distinct scenarios

scenario situation.

The safety metrics results are shown in Figure 24 based on the initial relative distance from the AV to NPC. It shows that the rate of collision safety violations for longer initial distances from NPC slightly increased while the success rate decreased. This is the only trend that can be identified from results for initial relative distance, so it can be concluded that speed is a more determining parameter for the safety testing of our AV.

Overall, the results in Figure 21 indicate that speed is a critical parameter for our AV safety testing platform.

Cybersecurity Test Case

For the cybersecurity test cases we chose 2 of the 15 distinct scenarios (Figure 21). This was to allow a greater scale of testing to be conducted on a select number of relevant scenarios. Scenario 10 was chosen as it demonstrated the most reliable performance, in terms of the most successful overtaking manoeuvres. Scenario 2 was chosen as it demonstrated the least successful results for overtaking. These two scenarios were run 50 times each, as had been conducted with the safety scenario runs. Figure 25 shows the performance of cybersecurity testing, conducted on scenario 2 and 10, in comparison to safety test cases.

Scenario 10 results reveal a discernible impact of the cyber attack. The LiDAR spoofing attack causes an increase in safety violations, prominently, in collisions and emergency breaking in the passing lane. This is also a concurrent result of the Scenario 2 test cases. Figure 17 shows the control level view, that incorporates sensor perception and mission and motion-planning. In the safety violation cases, we noticed that the euclidean clustering and kf_countour detect the spoofed LiDAR injection as an object and this false positive detection impacts the local-planning to force the AV to make the cut-in, in the overtaking manoeuvre process. Specifically, as the placement of the adversarial LiDAR device is on the left of the AV, the roll-outs of the left-side are blocked by the trajectory-evaluator.

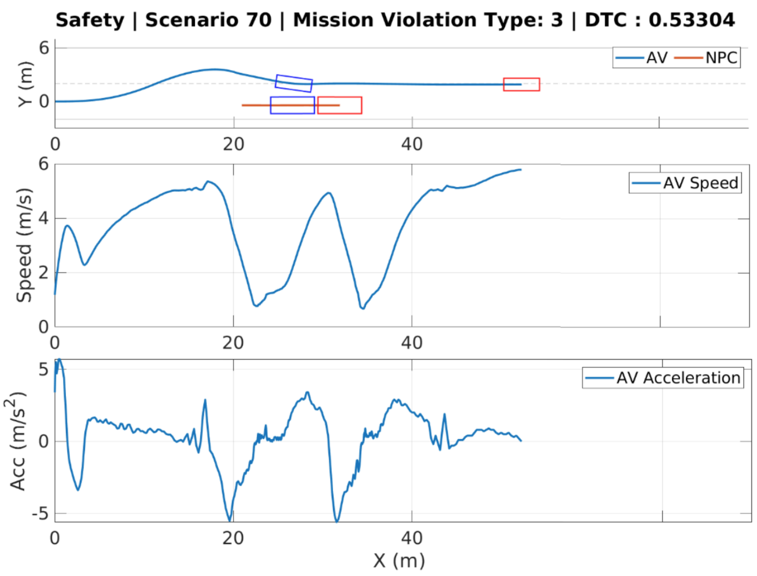


Figure 22: A Brake on Passing Lane safety violation

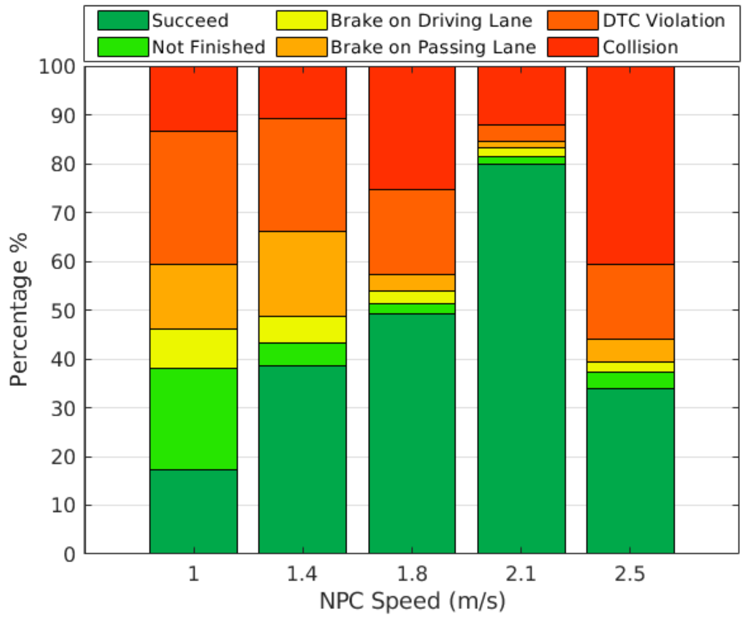


Figure 23: Test Results based on NPC Speed

This forces the AV to veer right and attempt the cut-in process that causes predominantly collision, DTC safety violations.

Cao et al. [35] and Hallyburton et al. [92] identify density of the spoofed points to be one of the key variables affecting cyber attack success rate. Figure 26 and figure 27 present the sensitivity of each attack parameter according to the cyber attack test cases. From evaluating the raw data of the test sets, and the sensitivity analysis for the cyber attack test cases of scenario 10, we concur with these assessments. We find the rate of

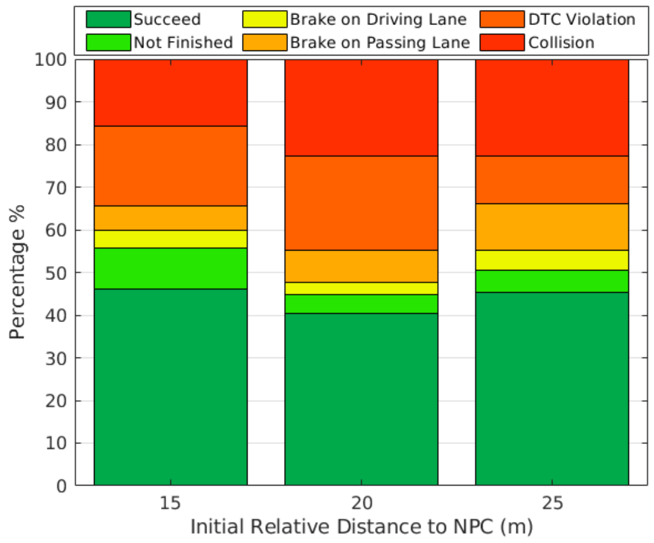


Figure 24: Results based on Initial Relative Distance to NPC

collisions is influenced by the density of the point cloud and the location of the attack. We can also see the influence the point of attack and duration have on causing a break on passing lane safety violation. As the duration of transmitting of the LiDAR point clouds increases and the location of the attack is further from the NPC, the likelihood of the AV initiating its breaks is higher.

In comparison, Scenario 2 cyber attack test case results show that safety violations are less sensitive to attack parameters. This can be due to the difficulty in interpreting the impact of cybersecurity on this scenario due to the already high rate of safety violations of the algorithms exhibited in the safety test case.

Table 16: Results of Cyber Attack applied to Scenario 10

Num.	V_{Col}	V_{DTC}	V_{BrP}	V_{BrD}	V_{NotF}	V_{Suce}
1	54%	20%	2%	0%	6%	18%
2	38%	38%	6%	2%	6%	10%
3	30%	28%	22%	2%	4%	14%
4	24%	28%	16%	6%	2%	24%
5	26%	16%	12%	6%	4%	36%
6	4%	4%	6%	4%	0%	82%
7	32%	14%	14%	6%	0%	34%
8	50%	24%	8%	2%	0%	16%
9	50%	30%	2%	2%	0%	16%
mean	34.2%	22.4%	9.8%	3.3%	2.4%	27.8%
std	15.9%	10.1%	6.7%	2.2%	2.6%	22.2%
min	4.0%	4.0%	2.0%	0.0%	0.0%	10.0%
max	54.0%	38.0%	22.0%	6.0%	6.0%	82.0%

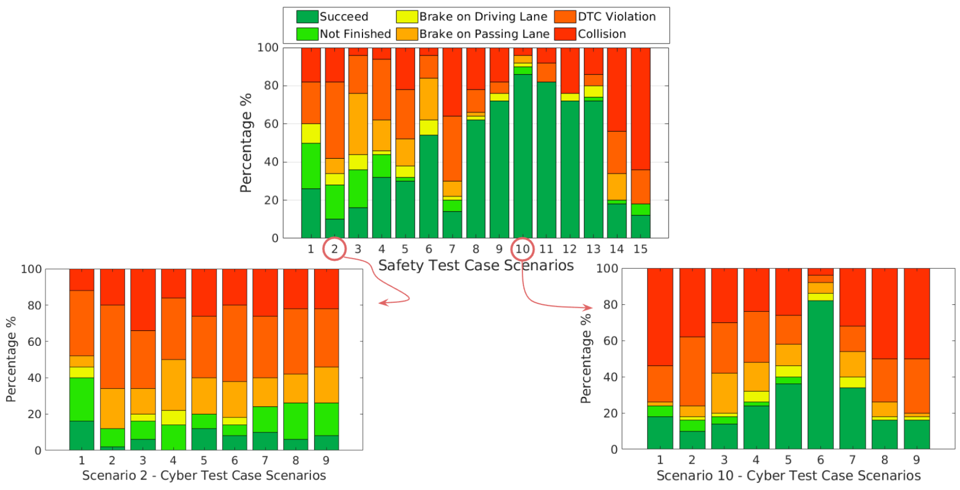


Figure 25: Performance Results Comparing Cyber Vs Safety Test Cases

Table 17: Results of Cyber Attack applied to Scenario 2

Num.	V_{Col}	V_{DTC}	V_{BrP}	V_{BrD}	V_{NotF}	V_{Suce}
1	16%	34%	28%	8%	14%	0%
2	26%	34%	20%	0%	8%	12%
3	20%	42%	20%	4%	6%	8%
4	26%	34%	16%	0%	14%	10%
5	22%	36%	16%	0%	20%	6%
6	22%	32%	20%	0%	18%	8%
7	0%	0%	0%	0%	0%	0%
8	0%	0%	0%	0%	0%	0%
9	0%	0%	0%	0%	0%	0%
mean	14.7%	23.6%	13.3%	1.3%	8.9%	4.9%
std	11.4%	17.9%	10.6%	2.8%	7.9%	4.9%
min	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
max	26.0%	42.0%	28.0%	8.0%	20.0%	12.0%

Real-World AV Testing

The real-world AV testing was conducted on a private road environment using our AV Shuttle, and an NPC vehicle (turquoise Mitsubishi iMIEV). The NPC vehicle is stationary during the tests as a safety assessment deemed it was too dangerous to conduct the experiment with a moving vehicle. This is due to the experiment being within a road environment where pedestrians and other vehicles are present. We conducted 3 test cases; a safety test case, cybersecurity test case and an optimised cybersecurity test case. The first test was an overtaking safety scenario. Two repetitions of the safety test case were conducted. The first test demonstrated a successful execution of the overtaking mission. The second test resulted in a DTC safety violation. The AV motioned to within 0.42 m of the NPC. The DTC violation is evident in Frame 3 of Figure 28, which details the second overtaking safety test case. Frame 4 demonstrates the eventual overtake after the DTC safety violation. Whilst the number of repetitions in the real-world pale in comparison to those conducted in the simulator, the real-world results concur with simulation results, that the

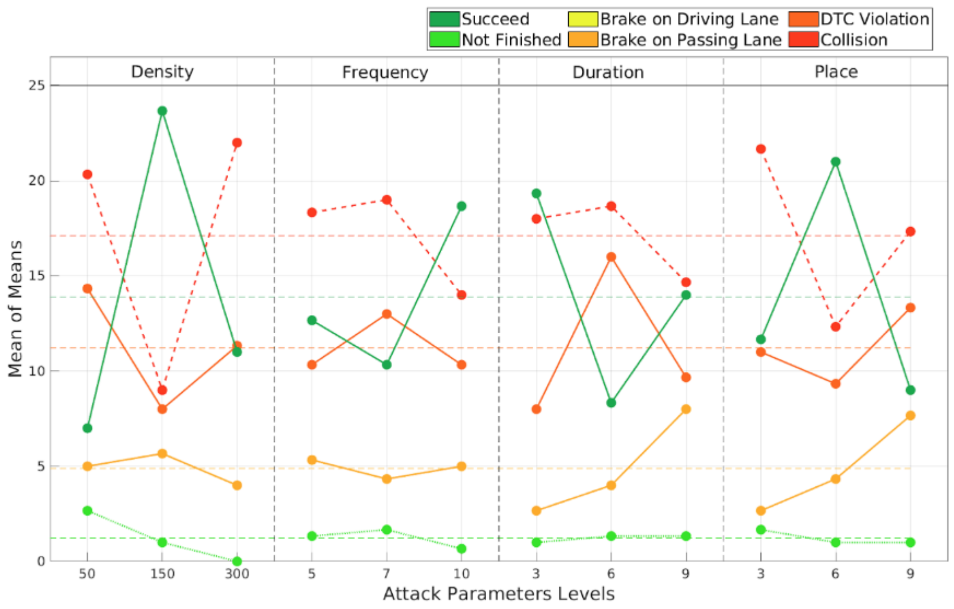


Figure 26: Scenario 10 - Cyber Attack Test Cases - Parameter Sensitivity

AD algorithm does not have enough reliability for the deployment in real-world missions.

Table 18: Result of the 3 real-world test cases

Test Type	Num. of repeats	success	Safety Violations
Safety Tests	2	1	1 DTC=0.42m
Cyber Tests	2	1	1 DTC=0.38m
Optimised Cyber Tests	1	0	1 DTC=0.32m

The cybersecurity test was conducted 3 times. Table 18 lists all the real-word experiments and their results. The first cybersecurity test demonstrated no impact from the spoofed LiDAR points and the overtaking manoeuvre was successful. The second cybersecurity test resulted in a DTC violation, the AV motioned to within 0.38 m of the NPC. After these two tests, we optimised the target angle of the spoofed points in relation to the attack scheme in Figure 75, to reduce the attack starting angle of θ_1. We did this because during the real-world test we observed that the reduced angle would provide assist the spoofed points to be closer to the AV trajectory and would cause the AV to detour from its intended route. It can be seen that this did work as the DTC decreased to 0.32 m. Figure 29 depicts the real-world cybersecurity test. Frame 2 represents the moment the attack was generated and perceived by the AD algorithm.

3.1.3 Discussion

From the analysis of the results we interpreted that different safety violations are connected to different modules of the AD algorithm.

Perception Module: We interpreted the cause of safety violations of the emergency break in the passing lane and emergency break in the driving lane to be related to the

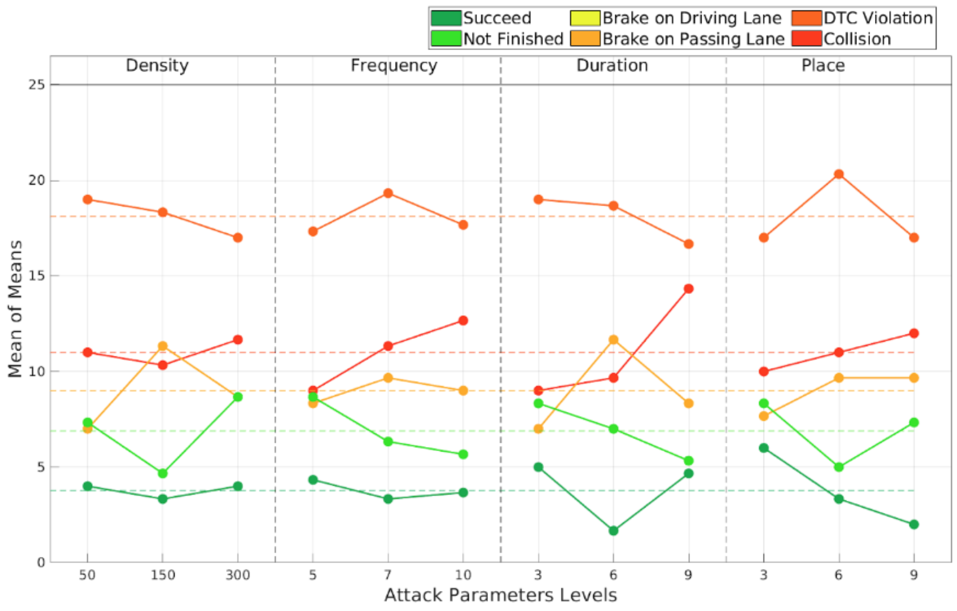


Figure 27: Scenario 2 - Cyber Attack Test Cases - Parameter Sensitivity

quality of the ground filtration. As we observed, ground filtering outcome changes during the AV manoeuvres (turns) because the shuttle body is tilted because of suspension and this results in the lidar reference frame orientation changing. Then some part of the ground point cloud as an unfiltered perception can be seen in the detection algorithms as an obstacle. This fake sudden obstacle might stop the AV during the motion. The spoofed LiDAR point cloud threat model is likely to make this condition worse. Optimisations for this: New body designs to rectify or limit the issues of LiDAR with the physics of the AV Shuttle are being developed. To focus specifically on these corner and edge cases and look at optimisation of the filtering of the perception algorithm. The latter recommendation is complicated by the fact it may include trade-offs; if the LiDAR perception algorithm is specifically tuned for this corner/edge case it could lead to over-filtration in normal driving scenarios, therefore this is one of the optimisation options to resolve the perception for the algorithm.

Open-Planner Module: We interpret the cause of safety violations for DTC and collision as due to an issue of the open-planner in predicting the trajectory of the NPC during the process of performing a cut-in, in front of the NPC. The optimisation would involve incorporation of features that would enable the prediction of the trajectory of the NPC and for perception improve the perception of the side-lidar to accurately perceive the NPC. We found that optimising all the perception and open-planner parameters for our shuttle model would significantly improve the reliability of the AD algorithm.

Open-Planner Developer Feedback

We sent a presentation of our results to the developers of the open-planner AD algorithm. In response, they acknowledged that it is a developing algorithm and we are engaged in more detailed discussions with them on how to optimise the algorithm. They also announced they are transitioning from Autoware.ai to Autoware.universe which is a more developed and advanced platform. Amongst their responses, they also pointed to the

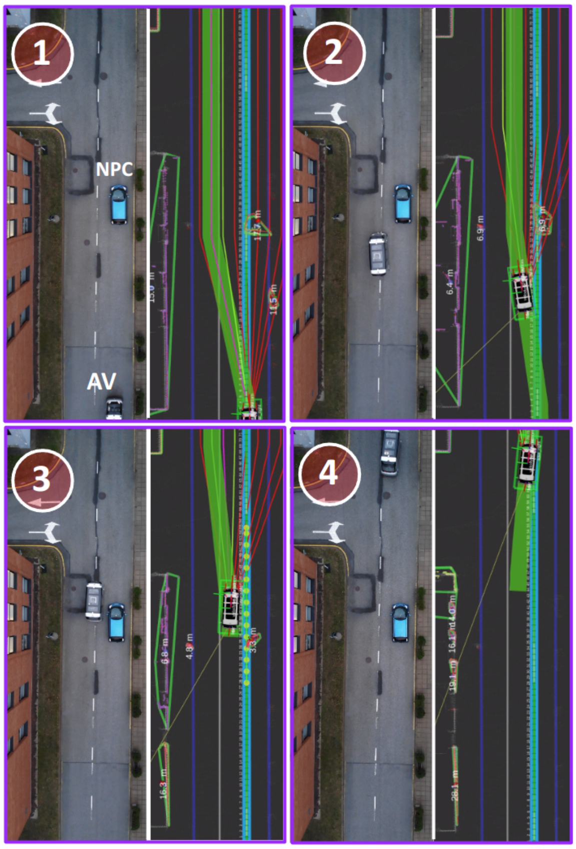


Figure 28: Real-World AV Test - Safety Test Case

novelty of receiving feedback on the reliability of cybersecurity test cases in addition to safety test cases.

3.1.4 Related Work

The closest contributions to our work are Yang et al. [325], Hallyburton et al. [92], Cao et al. [35] and Zhu et al. [341]. Each of these papers utilises a LiDAR spoofing threat model that varies based on the method for delivering the attack, adversarial generation and the type AD algorithm. Hallyburton et al. [92] target camera and LiDAR sensor fusion. They identify a blind spot between the camera and LiDAR sensor at the rear of the target AV. They use a malicious, 3D LiDAR point cloud array to inject malicious spoof points into the rear angle of the target AV. The attack was tested in a high-fidelity simulation and real-world against multiple perception algorithms. The results revealed a high rate of success utilising this attack. Cao et al [35], Yang et al [325], and Zhu et al [341] developed LiDAR spoofing attacks based on a threat model of a malicious LiDAR 3D point cloud injection in the road environment and by the roadside. Each of these contributions demonstrated that cyber attack results from AV simulation testing can be used to identify key parameters such as point cloud density, attack location and duration and that these parameters can be optimised to test the robustness of perception algorithms. We chose to extend from the related literature, in our work, in three areas; simulation testing configuration, safety criteria evaluation and target AD algorithm is in the developmental phase and is used within a real-world AV program. A feature of the selected work is that simulation

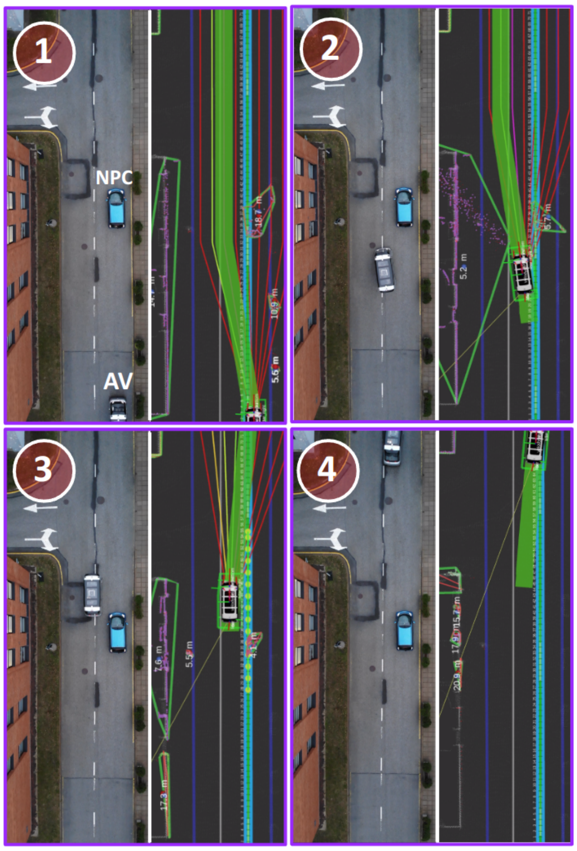


Figure 29: Real-World AV Test - Cyber Attack Test Case

testing often selected only one frame or a limited amount of frames and therefore the full driving mission was not observed. Whilst this is useful for reducing testing resource usage, running massive scale of tests and applicable to the scope of their work, as our study evaluates the AD algorithm and combines safety, our study focused on conducting simulation testing for the entire driving mission. Secondly, the evaluation of cyber attacks focused on attack success rate and attack parameters whilst the safety impact on the AV as a result of cyber attacks was not as clearly elaborated. In our study, we evaluate the cyber attack test cases with the same criteria as the safety case to derive the category of safety violation. Lastly, most of the simulations use default AV configurations and evaluate well-established algorithms. Our study uses a simulator configured for a real-world AV and evaluates an AD algorithm in the developmental stage where reliability and optimisation are required to be assessed under safety, non-cyber test cases before the impact of cyber attacks can be understood.

3.2 Analysing Adversarial Threats to Rule-Based Local-Planning Algorithms for Autonomous Driving

Navigation and planning algorithms are essential for AD. For the self-driving vehicle to navigate the road environment, the navigation and path-planning algorithm must calculate a route that ensures safety for the passenger and external environmental actors (pedestrians, other vehicles and road users, etc.) and achievement of the journey (mission).

Initial studies of navigation and path planning algorithms for AD have shown them to be vulnerable to adversarial attacks that introduce uncertainties into the route calculation, which causes downstream effects on the safe behavioral control of the AV. To improve the reliability of navigation and planning algorithms, they need to be further tested for uncertainties, and these methods are incorporated into the architecture of autonomous driving [34, 37, 334].

There are a few studies that focus on adversarial attacks on local-planning. These studies target machine learning algorithms for local-planning modules such as trajectory prediction (Trajectron++, Agentformer and GRIP++) [34, 36, 334]. The predominant threat model adopted, focuses on developing methods and tools of adversarial learning to understand the trajectory prediction model of the target AV and then either crafting malicious sensor data input or training other ego AVs in the driving environment to interfere with the target AVs predicted trajectory [34, 36, 303, 334]. The required result of a successful adversarial attack is to cause the target AV to generate a trajectory that is unsafe, inefficient, or uncomfortable for passengers. In this work, we expand on the target of attacks to a rule-based algorithm for local-planning, and focus on the trajectory generation and estimation of an AV. Our justification for focusing on rule-based algorithms is that, whilst AI approximate reasoning algorithms seem to be highly promising for the near future, an impediment to current adoption is the lack of feedback in real-world driving scenarios [52]. Rule-based algorithms for path-planning in robot navigation and AD are well-established, and more ubiquitous in real-world deployments.

A rule-based local-planning algorithm uses a cost function to estimate the least-cost path. The cost function takes input from immediately sensed-data; current pose, velocity etc.. The cost estimation is based on a calculation of factors such as; lateral collision, longitudinal collision, lane transition, central deviation etc., and weighting is given to these factors based on criteria such as safety and efficiency. By interpreting the cost-function, used for trajectory generation and estimation, as part of local-planning, an adversarial attack can be crafted which affects the downstream behavioral control whose decisions impact the safe driving state of the AV.

The main idea is that the white-box knowledge of the cost estimation function of the rule-based local planning algorithm can be used to craft adversarial attacks by manipulating factors inherent to the cost function. Evaluating white-box generated attacks enable an understanding of the level of stealth of the adversarial threat, and whether adversarial manipulation by the cyber attack can be distinguished from noise. Furthermore, these attacks will enable evaluation and assessment of the optimisation of the algorithm to uncertainties and the quality of decision-making.

The key questions we engage are the following:

1. What is the sensitivity of the cost function to adversarial data manipulation of key driving parameters?

2. How can an adversarial attack hide in the cost function from detection?

3. What optimisations of the rule-based algorithm can be considered to mitigate against adversarial data manipulation?

The problem area of this research, is centred on a local-planning algorithm, open-planner 2.5, which is used in an AV shuttle program that operates in real-world road conditions in Europe [57]. As with the open-source software community, development of vulnerability research and testing methods proliferate across the ecosystem and are utilised and innovated for diverse platforms. The aim of this study is to focus on the vulnerability of the local-planning function of autonomous driving and provide direction and guidance to the autonomous driving security community to develop vulnerability testing on diverse planners and algorithms. In a broader sense, this research aims to understand how AD algorithms used in real-world AD programs can be tested for adversarial threats and validated to improve assurance for real-world operational driving.

3.2.1 Threat Model

The attack targets the local planning cost function, with the aim of inducing the trajectory evaluation to choose a motion-planning route that is not optimal for safety, functionality of the driving mission and comfort of the passengers. To achieve this, the most direct mechanism to impact the cost function is to manipulate, with adversarial data, the sensed-data input that is inherent to local-planning. The Current_Pose data is the optimal target for this as it is the primary sensed-data for localisation of the vehicle, containing the longitudinal, lateral positioning and orientation of the AV. Whilst altering the pose data of the vehicle has previously been conducted in other studies [34, 36, 303, 334], in our attack we aim to explore the sensitivity of our cost function to data manipulations and conducting the attack during specific time-intervals.

For the threat model used in our study, we assume that the attacker has access to the internal network of the AV and is able to listen to control message communications and collect data. This could be achieved through supply-chain compromise of a library in the control software, insider threat actor, or many of the vulnerabilities in existing communication frameworks for autonomous systems such as the robotic operating system (ROS) [64]. Given the attacker has access to the internal network, the question arises, why not change the Lane_ID or a driving parameter which would be more simplistic and direct? We view these attacks as overt in nature and likely to be detected, the compelling nature of adversarial data manipulation is that the attack is difficult for AV safety engineers to interpret between noise and an explicit cyber threat. Another consideration are the external interfaces of the vehicle localisation sensing, which generates the pose data. It is a possibility that the pose data can be manipulated by an external attack in the form of GPS spoofing or an adversarial LiDAR, dependent on the sensor configuration used for the localisation of the vehicle. The study focused on the vulnerability of the planner and its search space, considering localisation. We considered internal attacks to be important due to the increase in attacks through software and hardware supply-chains, and therefore the scope of the attacks within the study highlighted this area.

Attack Case 1: Position Offset Attack

The attacker creates a spoofed ROS topic which is able to deliver malicious input data of the Current_Pose (longitude, latitude, and velocity) to all the nodes of the local planning module. The data manipulation is injected online/dynamically during the critical overtaking manoeuvre involving the AV and NPC. Figure 30 displays the critical driving scenario and the time frames in which the manipulated Current_Pose data is injected into the local planning pipeline cost estimation. The red dashed lines in Figure 30 represent the roll-outs, and the green highlighted, denoting the selected motion-path.

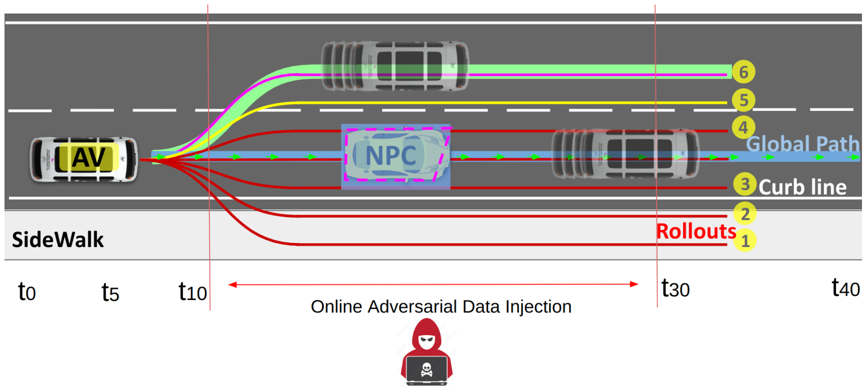


Figure 30: Threat Model

For the manipulation of the Current_Pose data, we introduce a deviation to lateral and longitudinal pose. For the lateral pose data, the sensitivity deviation introduced was structured as follows:

- Attack Case 1a: 0.16%
- Attack Case 1b: 0.33%
- Attack Case 1c: 0.5%

In designing the range of deviation, we considered state-of-the-art attacks such as AdvDO attack [34], which noted two requirements for developing adversarial threats to planning algorithms:

1. Malicious data input needs to be feasible to the real, physical constraints of the vehicle [34].
2. Malicious data input of the local-planning algorithm should be close to the nominal trajectory [34].

Therefore, we chose a range from a slight perturbation of pose to a 1m deviation. The longitudinal pose data sensitivity deviation range was structured as follows:

- Attack Case 1d: 0.33%
- Attack Case 1e: 0.66%
- Attack Case 1f: 1.00%

This range is the same as the longitudinal deviation. The difference in percentage comes from the difference in coordinate values of lateral and longitude. The lateral value is almost double those of the longitudinal, and therefore the percentage is doubled.

Attack Case 2: Message Time-Delay

For the second attack case, we inserted a time-delay into the messages of the Current_Pose topic communicating to the nodes of the local planning module. We introduced a message delay when the AV passes 2m in front of the NPC (from the centre) in the lateral direction. We introduce 3 different time delays in the message:

- Attack Case 2a: 0.3 seconds
- Attack Case 2b: 0.6 seconds
- Attack Case 2c: 1.0 seconds

The message frequency is approximately 50hz, so this is a message every 20 milliseconds. We chose the above range of deviation of time-delay as it enabled a spectrum of a message from the delay from approximately 15, to 50 messages.

3.2.2 Experimental Setup

Test Environment and Configuration

In terms of conducting such experiments, simulation is the best method among all testing methods for AVs. To accelerate the testing, we bypassed the sensing and detection nodes of the algorithm and focused on the planning part by utilizing the low-fidelity simulation feature provided by Autoware.ai and Openplanner. The low-fidelity simulation uses the open-planner 2.5 control algorithm. It provides simulated localization and detection data for the planning nodes and receives the actuation commands to simulate the AV kinematics. This process runs faster due to the low-detail environment required for the simulation and the lack of the process to simulate the sensors. Figure 31 displays the different frames of an overtaking simulation in the simulator.

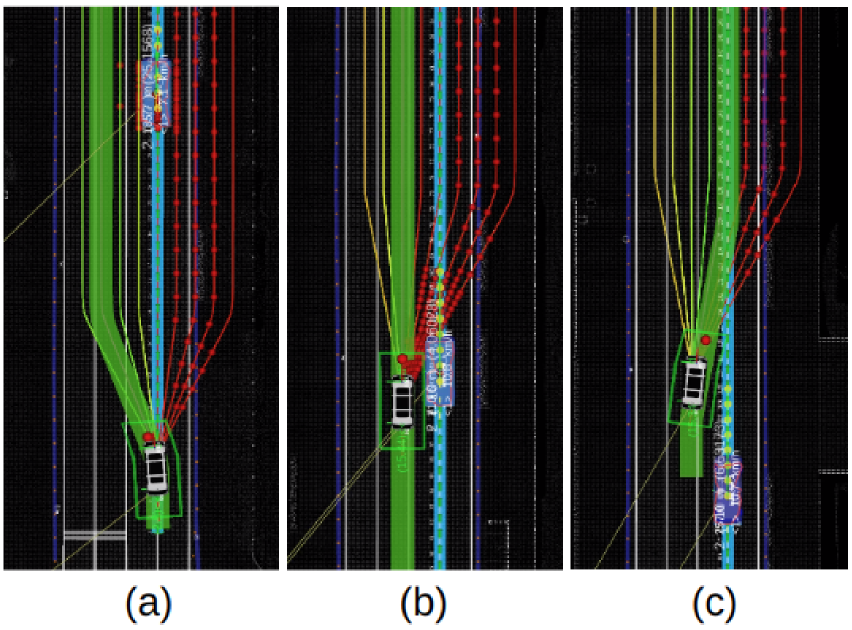


Figure 31: Example of an overtaking simulation in the low-fidelity simulator, a) starting point of the overtaking b) middle of the mission, AV is on the opposite lane reaching the NPC c) AV cuts in

Target Mission

Overtaking is one of the most challenging maneuvers for AVs [186]. In this research, we selected this operation as the target scenario for studying the planning algorithm under the cyber-attack. The scenario parameters in Figure 32 are listed in Table 19.

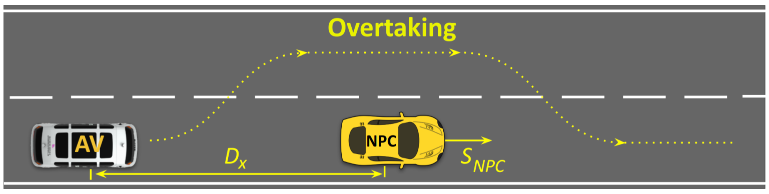


Figure 32: Overtaking Scenario and parameters

Table 19: Target scenarios definition

Actor	Speed (m/s)	$D_x(m)$	Goal
AV	[0:6]	0	overtake the NPC safely
NPC	3	25	keep moving

Safety Evaluation Test

To assess the safety and reliability of the planning algorithm in normal conditions (no attack), we ran the scenario simulation 300 times to reach a meaningful statistical population. Then, the planning algorithm behavior in each case was evaluated with the local-planner performance evaluation criteria (explained in the next section).

Attack Test Cases

Finally, the platform was used to simulate the proposed adversarial data manipulations and time-delay messaging, during the overtaking mission and monitor the algorithm's behavior. For each attack case, we ran the simulation (with attack) 100 times. Overall, 900 simulations were conducted for all attack cases.

Evaluation Criteria

For the evaluation, we used previously established safety criterion [190] with evaluation criteria recommended by SafeBench, a benchmarking framework for safety evaluation of AD algorithms for critical driving scenarios [320]. Table 20 displays the metrics used for the performance evaluation.

Table 20: Safety Evaluation Criteria

Safety Condition	Data Label	Description	Metric
Succeed	Suce	AV Successful complete the mission	Pass/Fail
Not Finished	NotF	Failure to finish the mission	Pass/Fail
Distance-to-Collision	DTC	Violation of the safe distance between AV and NPC	AV within 0.5m of other vehicle
Break on Driving Lane	BrD	AV initiates emergency break on driving lane	Pass/Fail
Break on Passing Lane	BrP	AV initiates emergency break on passing lane	Pass/Fail
Collision	Col	AV collides with NPC	Pass/Fail
Violation	V	Safety Violation	

3.2.3 Results

After running 1200 simulations, all recorded data including the AV and the NPC position and orientation were processed to assess the simulations based on the evaluation criteria. We also visualized the recorded data to study the violation and their cause in each simulation as shown in Figure 33. Figure 33.a represents a safety run completed successfully. Next, (b) and (c) display lateral and longitudinal attack cases which experienced brake and collision safety violations respectively. Finally, (d) shows a message time delay attack which is finished by a collision. The asterisk signs in the AV trajectory show the point where the Openplanner changes the rollout. Overall, all the safety violation results for the whole experiment are presented in Figure 34.

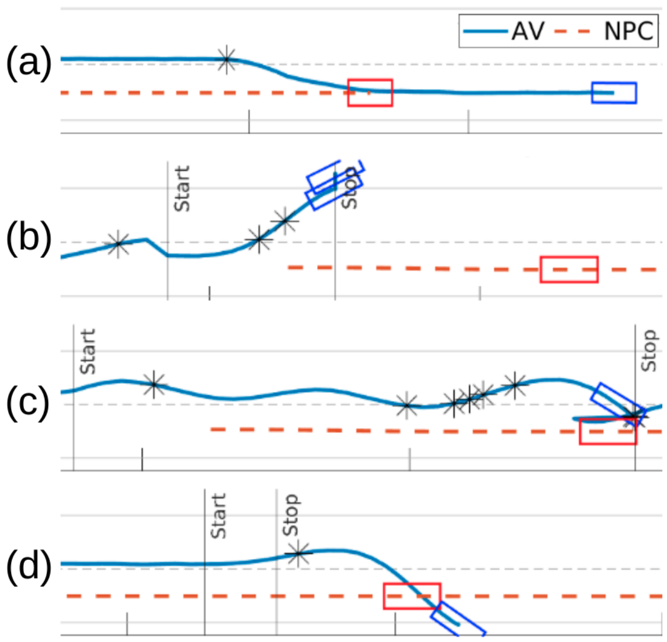


Figure 33: 2D representation of the simulation of each test group. a) a successful safety test, b) a lateral attack case that led to a brake violation, c) a longitudinal attack case that experienced a collision, and d) a message time delay that causes a collision. for the attack cases a vertical line shows the start and stop point of the attack

For each of the attack test cases, we saw an increase in safety violations of the AV compared to the normal safety test case experiment. As the value of the deviation for lateral and longitudinal values increased the number of successful mission completions decreased. Although marginal, the greater number of safety violations for the attacks on the Current_Pose data were observed in the lateral deviations. Given the importance of lateral positioning to the overtaking manoeuvre, this can be understood as any deviation increases the complexity of executing the overtaking manoeuvre. In the 1f attack test case, the highest value longitudinal change (approximately 1 meter) led to a crash with curbside and not able to continue the mission. This event was reported as a braking safety violation.

The time-delay messaging attack test case saw the only result for mission not finished metric. Furthermore, the greater the delay of the Current_Pose data reaching the local-planning nodes, the increased likelihood that a safety violation will occur, and in the case of our experiments, the greater the likelihood of a the most serious safety violation, collision.

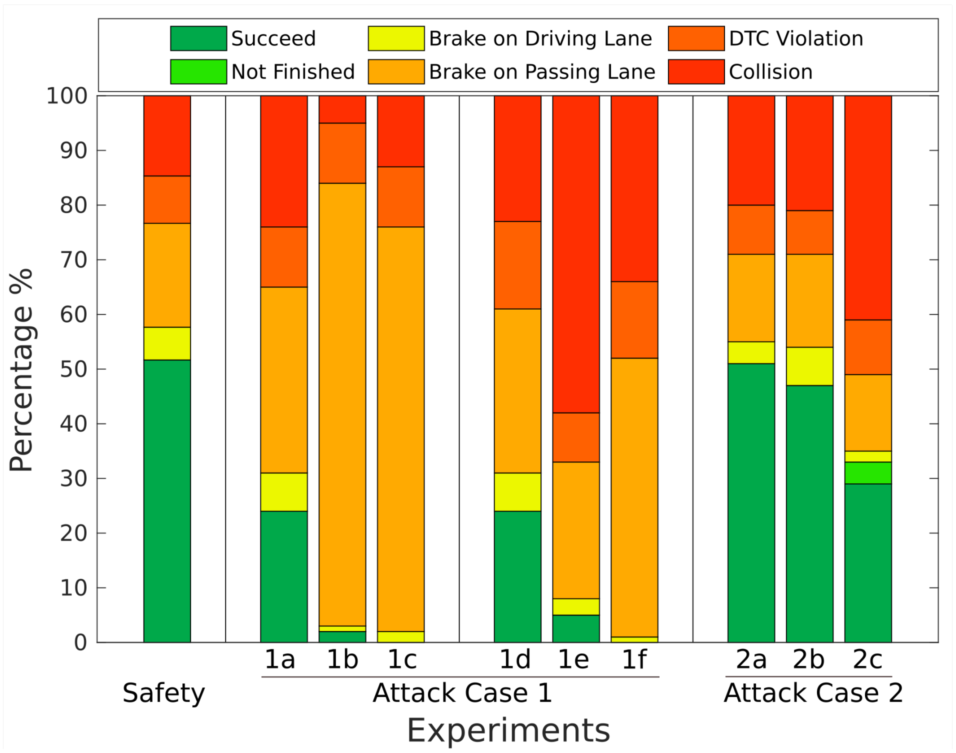


Figure 34: All simulation result based on the proposed safety criteria

Table 21 demonstrates the results of the safety test according to the performance evaluation criteria. The level of safety violations are reflective of an algorithm which is in development and being optimised for critical driving scenarios such as overtaking.

Table 21: Summary of the Safety Simulation

Num.	V_{Col}	V_{DTC}	V_{BrP}	V_{BrD}	V_{NotF}	V_{Suce}
300	4.6%	8.6%	19%	6%	0%	51.6%

	TS	ACC	YV	LI
mean	29.1	0.4	3.8	7.1
STD	6.7	0.2	2.2	4.6
min	21.9	0.2	1.8	2
max	42.3	1.3	21.7	25

Table 22 shows that for each deviation there is a high number of safety violations in comparison to the safety test case results. In regards to the sensitivity analysis, a smaller deviation of around 20 to 25 cm can achieve the result that the local-planning algorithm is only successful in generating a trajectory that completes the mission in 24% of the total test set. Furthermore, a small deviation in the lateral pose, can achieve a higher number of

Table 22: Summary of the Attack Case 1: Position Offset Attack Simulation

Case	Num.	V_{Col}	V_{DTC}	V_{BrP}	V_{BrD}	V_{NotF}	V_{Suce}
1a	100	24%	11 %	34%	7%	0%	24%
1b	100	5%	11 %	81%	1%	0%	2%
1c	100	13%	11 %	74%	2%	0%	0%

1a		TS	ACC	YV	LI
	mean	35.3	0.4	9	7.5
	STD	7.4	0.2	7.5	5.4
	min	21.9	0.2	1.9	1
	max	42.4	1	23	23

1b		TS	ACC	YV	LI
	mean	41.4	0.4	9.5	4.8
	STD	3.5	0.1	4.4	3
	min	22.1	0.2	3.1	1
	max	42.4	1.2	23.7	21

1c		TS	ACC	YV	LI
	mean	41.7	0.4	7.8	4.7
	STD	1.7	0.1	1.2	2.7
	min	32	0.3	4.3	1
	max	42.3	1	9.8	15

collisions with an ego vehicle. It may also be seen from the lane invasion and steering angle results that small deviations to lateral pose result in a fluctuation of the cost of different rollouts which cause greater lane transitions as the cost function causes the AV to choose a route based on minimum cost. The higher deviation results in a higher occurrence of breaking activity and hitting the curb. Furthermore, the higher deviation results in the AV being stuck in the passing lane, this is due the dramatic change in lateral pose. The 1 meter deviation attack case results in 0% success of finishing the mission.

Table 23 results of the longitudinal deviations also display a high number of safety violations in comparison to the safety test case results. Collision safety violation is highest for the longitudinal deviation attack. This can be reasoned as the longitudinal deviation does not experience the same high volume of breaking passing lane safety violations, where the vehicle gets stuck, as seen with the lateral pose deviation. The higher deviation of longitudinal pose, results in increased acceleration and this causes sharp breaking. This is indicated with the 1f result, the 1 meter deviation attack case, which displays a higher instance of breaking safety violation. The 1 meter deviation attack case results in 0% success of finishing the mission.

Table 24 demonstrates the shorter delay of local pose data has minimal impact on the success of the mission and safety violations. As the time duration of the message delay

Table 23: Summary of the Attack Case 1: Position Offset Longitudinal Deviation Simulation

Case	Num.	V_{Col}	V_{DTC}	V_{BrP}	V_{BrD}	V_{NotF}	V_{Suce}
1d	100	23%	16 %	30%	7%	0%	24%
1e	100	58%	9 %	25%	3%	0%	5%
1f	100	34%	14 %	51%	1%	0%	0%

1d				TS	ACC	YV	LI
			mean	33.8	0.5	5.7	9.1
			STD	7.6	0.3	4.9	5.4
			min	18.1	0.2	1.7	2
			max	43.2	1.4	23	27

1e				TS	ACC	YV	LI
			mean	32.2	0.6	6.7	10.5
			STD	9.5	0.2	3.2	5
			min	17.8	0.2	1.9	2
			max	43.2	1.1	20.5	25

1f				TS	ACC	YV	LI
			mean	32.2	0.7	5.9	11.3
			STD	7.9	0.2	2.5	4.7
			min	18	0.3	2.7	2
			max	43.2	1.4	22.1	26

is increased the impact to the reliability of the local-planning algorithm is higher. Test 2c, which is the delay of Current_Pose data of 1.0 second, shows considerable increases in collisions and decreases in the likelihood of the success of the mission. The time-delay of the pose data to the local-planning nodes results in a loss of localisation and the greater delay the greater impact on the cost calculation which in turn causes uncertainty for the behaviour selector/decision-making.

3.2.4 Discussion

The results of the test simulations demonstrated that the cost function is sensitive to minor deviations of both the lateral and longitudinal pose. The success rate of the mission is visibly diminished when adding adversarial data manipulations to the sensed-data input. The higher the deviation, the higher the likelihood of mission failure. The minor deviation attacks, where the deviation is a range of 20 to 25cm offer a good starting point to mutate adversarial data for further attacks based on this range. Whilst the higher range attacks conducted in our experiments showed a higher rate of mission failure, a deviation of 1 meter can be seen a noisy enough to be observable. We also noticed such behaviour in a real-world AV shuttle [254] and a manual emergency break had to be enacted to prevent an emergency.

Table 24: *Summary of the Attack Case 2: White-Box Delay Simulation*

Case	Num.	V_{Col}	V_{DTC}	V_{BrP}	V_{BrD}	V_{NotF}	V_{Suce}
2a	100	20%	9 %	16%	4%	0%	51%
2b	100	21%	8 %	17%	7%	0%	47%
2c	100	41%	10 %	14%	2%	4%	29%

2a		TS	ACC	YV	LI
	mean	29.3	0.4	4.2	7.6
	STD	8.1	0.2	2.2	5.4
	min	18.1	0.2	1.8	2
	max	53	1.1	16.7	24

2b		TS	ACC	YV	LI
	mean	30.6	0.4	4.8	7.8
	STD	8.6	0.3	3.7	4.8
	min	22.9	0.2	1.8	2
	max	58	1.1	23.8	21

2c		TS	ACC	YV	LI
	mean	32.9	0.4	7	8.3
	STD	9.6	0.3	5.2	5
	min	13	0.2	1.1	0
	max	58.2	1.3	22.9	23

The time-delay attack demonstrated that minor delays cause minimal impact on the success of the mission and the occurrence of safety violations. Delays in sensed-data input flowing to the local-planning modules of greater than 1 second increase the rate of mission failure and safety violations. Given that 1 message is broadcast every 20 milliseconds, 1 second represents around 50 messages, and a delay of this magnitude is also likely to be more observable.

For the attack to hide in the cost function, investigating mutations for minor deviations of lateral and longitudinal values in the range of 20 to 30 cm, offer an optimal target range.

Mitigation of the adversarial deviation and time-delay attack could include the implementation of a redundant driver. This means that the AV should run a concurrent process executing a concurrent planning instance. If the redundant driver and the actual driving algorithm give different results, then this could indicate that an attack might be happening. In such a case, the AV could either stop safely awaiting for human intervention or switch to the redundant driver to complete its mission. The development of the architecture for a redundant driving integrity checking function also needs to consider isolation from the primary driving function so that an attacker cannot also compromise both.

3.2.5 Relation to existing solutions

As safety validation of AD algorithms is a critical field for the adoption of AD in real-world environments, there is a focus on testing the reliability of trajectory prediction and generation to adversarial driving actors in the road environment. Wang et al. [303], Abeysirigoonawardena, Dudek & Shkurti [6], Chen et al. [44], Klischat et al. [157], and O'Kelly et al. [215] use simulation environments to develop adversarial trained NPCs whose driving actions cause safety violations of the trajectory prediction of the targeted AV. These simulations are focused on safety validation and are not focused on the exploitation of the algorithm by adversarial threat actors, however, their methods in generating adversarial examples and target parameters and data values are of great use in developing adversarial cyber threats.

On a practical level, involving the real-world operation of AVs, there are few research studies into the robustness of planning and navigation algorithms to adversarial threats. Prominent among them are Zhang et al. [334], Cao et al. [36] and Cao et al. [34]. These studies focus on the robustness of the trajectory prediction, the ability of the AV to predict the trajectory of another ego vehicle or environmental object (pedestrian, animals etc.) and make driving decisions accordingly. The attacks in these studies are targeted at deep-neural networks (DNNs), and therefore focus on adversarial learning to develop robust adversarial trajectories. In relation to our work, the observations on ranges for deviation of lateral and longitudinal values and the considerations for crafting adversarial data were useful in developing our attack cases.

In this experimental research, we conducted a sensitivity analysis of the openplanner 2.5 rule-based planning algorithm to adversarial data manipulation of lateral and longitude values and delayed sensed-input messages to local-planning nodes. We evaluated these attacks in a low-fidelity simulation test environment using an overtaking manoeuvre critical driving scenario. The results showed that the planning cost-function is sensitive to adversarial data manipulation that introduces deviations to the lateral and longitudinal values. These adversarial deviations cause higher rates of failure to complete missions and cause safety violations. For the message delay attack, limited delays in the range up to approximately 0.6 seconds have a limited impact on the trajectory calculation. Message delays for 1 second or greater cause a visible difference in the safety violation rate and mission success. We opine that limited deviations are an optimal area to explore further attacks and in more diverse critical driving scenarios.Through this work we propose a class of stealthy attacks on the local-planning function of AD. An area of future research is the development of monitoring systems developed around such basis of attacks. The results show the feasibility of monitoring real-time properties of the messages propagations and therefore post-mortem forensics might be able to determine the presence of an attacker causing safety violations of AVs.

3.3 Analysis of Autonomous Driving Software to Low-Level Sensor Cyber Attacks

Cyber attacks which manipulate input to physical processes in cyber-physical systems present a fundamental challenge to secure system design [331]. Within the domain of automotive systems, transformation of legacy, analog architectures to digitally connected and AD technologies present new challenges. Legacy, analog automotive systems were designed based on a principle of contained, isolated system boundaries, restricting the flow of data within an analog system and sub-system [30]. The AD system architecture transforms this design, requiring the lower-level, analog control of actuation processes (steering control, braking, acceleration etc.) to be open and connected to digital controllers so their process signals can be translated to digital input for the higher-level decision control [205].

There have been numerous real-world examples of semi-autonomous control architectures enacting unsafe decisions from erroneous sensing data from low-level actuation sensors [294] [74]. The 2018 SmartLynx Airline incident demonstrated that a physical disturbance from a maintenance activity on the horizontal stabilising sensor caused the sensing input to send erroneous data which propagated through to the control systems for flight planning, stabilisation and safety. The control systems initiated multiple concurrent actuation decisions (horizontal stabilisation, acceleration etc.) which affected the safe operation of the flight [74]. Ultimately, manual intervention to override the autonomous control resolved the unsafe state of the flight.

Within the context of cyber threats, numerous studies have proven the vulnerability of microelectronic sensors to electromagnetic interference (EMI) [225], [290], [336], acoustic sensor [289] [283] and data manipulation attacks [60], [131], [202], [51]. Furthermore, the network that exchanges actuation signals, CAN Bus network, has been shown to be inherently vulnerable to a diversity of man-in-the-middle [31, 139] attacks. Yet, there is a lack of practical investigation which extends this analysis of the propagation of malicious data input within an AD system, where physical processes are software controlled and manual, human intervention is not available.

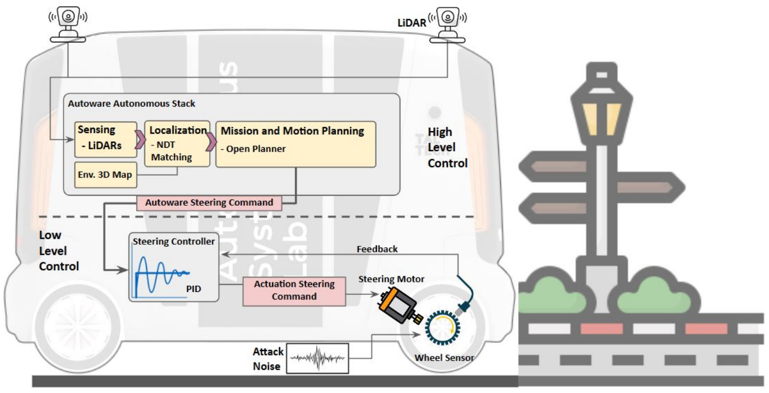


Figure 35: High-level architecture of Steering Angle Sensor Manipulation within AD System.

This experimental research is motivated to investigate how cyber attacks to electromechanical components, in our case, a steering-angle sensor, propagate through the AV system, affecting higher-level decision-making. The aim of this research is to analyse the design of a real-world AD vehicular system and assess mechanisms to enhance the design of the architecture of AD systems to be more robust and resilient. To achieve this, we, firstly, investigate a real-world AV software ecosystem, analysing the integration between the lower-level control, characterised by electromechanical components, and the high-level control, characterised by digital systems which support algorithmic decision-making. Secondly, how malicious input propagates within this ecosystem. Finally, determine mechanisms for enhancing secure design.

To guide this research, we focus on the following research questions:

RQ1 *How does a manipulation to the electromechanical component propagate through the AD software stack?*

RQ2 *What dependencies exist between the AD control algorithm and low-level control?*

RQ3 *Where in the architecture of the autonomous vehicle can defensive mechanisms be placed to defend against control invariants?*

3.3.1 Approach Overview

Our approach (see Figure 36) is to, *firstly*, implement the sensor interference attack model in our custom high-fidelity AD test-bed environment. The test-bed environment contains the software stack of our real-world vehicle and configurations consistent with the real-world kinematics of the vehicle.

Secondly, from the results of the experiments, we assess the impact of the cyber attacks utilising defined safety criteria. Furthermore, we conduct a sensitivity analysis of the vehicles dynamic parameters to identify the behavioural affect of the malicious input and assist in pinpointing critical areas of the AV software which are affected by the attack.

Third, we conduct a bottom-up analysis, to ascertain what happens to the high-level, decision-control, when malicious data is injected into the low-level. The bottom-up analysis details the relationship between inputs and outputs in the AV software stack.

Fourth, the previous analysis enables backstepping at a conceptual level to stabilize elements of the control model which are susceptible to the sensor interference attack.

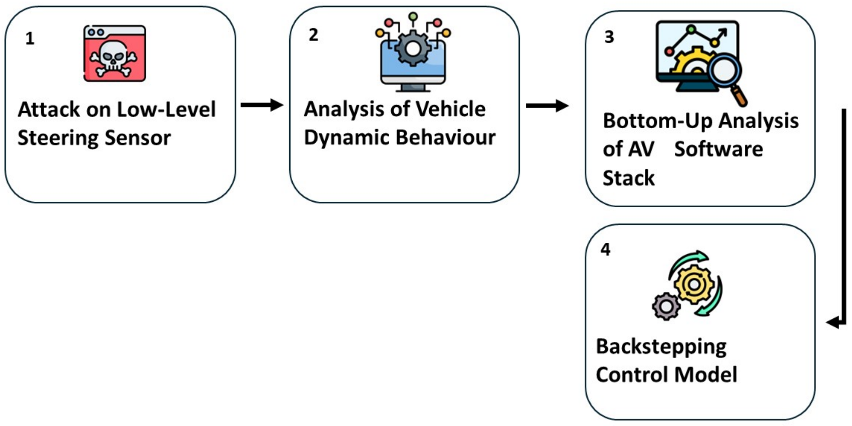

1
Attack on Low-Level Steering Sensor
2
Analysis of Vehicle Dynamic Behaviour
3
Bottom-Up Analysis of AV Software Stack
4
Backstepping Control Model

Figure 36: Conceptualization of our approach, from attack to backstepping.

We justify the use of this approach as it enables us to take an architectural view of the AV software stack. Existing studies use methods that view the problem of manipulation of low-level sensor input either within the context of a PID control [225] [290] issue or solely focus on the autonomous control [131]. We believe, taking an architectural perspective, where the interconnections and dependencies of the system are encountered, enables the designer/s of the AV to gain more insight into the functioning of the system under attacks.

3.3.2 Adversarial Model

The objective of the attacker is to cause the AV to take unsafe driving actions resulting from manipulation of the steering angle sensor. We assume the attackers cannot directly access the digitised sensor readings. Instead, we assume that the attacker can exploit vulnerabilities in the steering angle sensor using proven techniques such as EMI, to affect the integrity of the sensor data (analog signals on the signal conditioning path before being digitised).We assume that the attackers can physically place an EMI device near the steering angle sensor and are capable of crafting and transmitting interference to the sensor during the navigation of the AV and thus transform the waveform of the sensor output. We further assume that the attackers do not possess an in-depth understanding of the voltage levels of the steering sensor and therefore focus on injecting incremental noise into the sensor. We assume that the attackers can observe the operation of the AV and control the attack in terms of initiation and cessation of the attack during varied time periods or within the frames of a critical driving manoeuvre.

3.3.3 Attack Model

The attack is conducted in the measurement of the input and output of the PID controller for the steering angle (See Figure. 37).

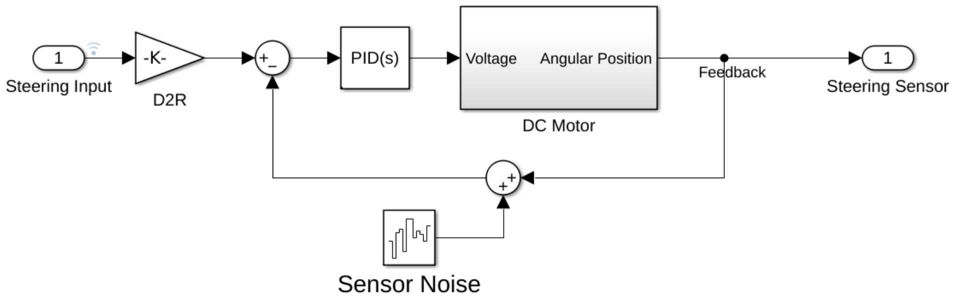


Figure 37: Steering angle sensor attack.

The key parameters that affect the success rate of the attack are: *duration*, *noise*, *attack trigger action*.

Within, our attack model, attacks are conducted with differing sensitivity levels of the steering angle sensor, durations and are triggered at targeted points of the AV mission. We have chosen a range of sensor attack noise levels (0.01, 0.05, 0.1, 0.2), rather than a specific target point. We expect that our attack, will generate errors that propagate from the low level to the localizer and trajectory-generator blocks. The study of Pöllny et al. [225], which conducted EMI attacks on a sensor used in an automotive ECU, indicated that an attacker does not need to set a specific value for the steering angle attack, but simply to find the sufficiently high level of noise that would alter system behaviour to the attacker goal.

Whilst, EMI attacks have been proven successful against microelectronic components in [150] [131] [225] [60] [290] [336] [283], the attacks are applied to the stand-alone sensor hardware and application use-cases such as microphones, temperature sensors, drones. The novelty of the attack model in our study is the implementation of the attack to a fully-autonomous vehicle that integrates low-level actuation with high-level AD decision-making. This enables the ability to assess the affect of the attack to the entire AV software stack. Furthermore, the attack is conducted utilising scenario-specific testing. This is of critical importance, as it is widely understand that the performance of the AD decision-making layer differs based on scenario specific behaviour [131]. For the AD algorithms may be better optimised for specific driving manoeuvres such as overtaking, or ODDs such as busy intersections. Our attack is conducted in a simulation test environment, as attacks at the physical, hardware-level are proven, the gap in existing research, is how these inputs propagate within the system and affect the decision-making within an autonomous system.

3.3.4 Experiment

Experimental Setup

To conduct the attack and analyse the subsequent effects, we developed an experimental test environment.This environment consists of a simulation platform that fuses the low-level actuation, simulated in MATLAB, with a high-fidelity simulation of the AV software of our real-world vehicle, simulated in CARLA. The simulation test environment provides an optimal platform as it uses the same mathematical model of the steering actuation sensor and the same software stack as the real-world vehicle. Furthermore, the simulation environment enables attack testing to be conducted in an agile manner, whilst, removing the safety risk factors of testing the AV in the physical, road environment.

Attack Implementation

We chose to conduct the low-level attack on three diverse scenarios (see Figure. 38): 1) Straight-line, 2) Overtaking manoeuvre and, 3) Left-turning maneuver at intersection. These scenarios were chosen as they are consistent with the most-popularly tested driving scenarios according to the survey of test methods and practices by Lou, Deng, Zheng, Zhang & Zhang [177]. As shown in Figure 38, the high-fidelity simulation view for the 3 scenarios is conducted. The Straight-Line scenario shows that the EMI attack is initiated after the vehicle traveled 20 meters, with two different attack durations: 10 and 20 meters. For the overtaking manoeuvre, the attack begins during the cut-in process and lasts for 10 meters. Finally, in the intersection scenario, the attack is launched as the vehicle enters the intersection and persists for a distance of 10 meters.

To conduct our experiments, firstly, we conduct the scenario with *no-attack* for 100 runs. This establishes a baseline of the performance of the AV without attacks. From there, each of the attacks with different noise levels and duration are run 100 times. Overall, approx. 1900 simulation runs are recorded, and as the high-fidelity simulation uses GPU and CPU resources, this is a time-consuming process. Figure 39 presents the scenario flow used to integrate the attack into the mission in CARLA. It outlines the sequence of behaviors from the vehicle's initialization and driving towards the goal to executing an attack or stopping based on a distance trigger. The attack is enabled based on a predefined condition. This structured flow allows for precise control over when and how the attack occurs during the scenario, ensuring consistent testing of the AV's response to disturbances.

Evaluation Criteria

Table 25 and 26 detail the safety and performance criteria applied in our experiments, respectively. As we have diverse scenarios which involve scenarios with ego vehicles, certain criteria is only applicable to their corresponding scenario. In this analysis, mission failure (NotF) and safety violations (SafetyV) are distinct evaluation criteria used to assess the performance and safety of the AV during the scenarios.

Mission failure (NotF) refers to instances where the vehicle was unable to complete the mission. This typically occurs due to critical events that prevent the AV from finishing its task, such as collisions (V_{Col}), localization loss (V_{NDTLs}), or sidewalk incursions (V_{SiIn}). These violations are severe enough to terminate the mission.

Safety violations (SafetyV), on the other hand, refer to any breaches of safety that occur during the mission but do not necessarily prevent the vehicle from completing it. A mission may still be considered successful even if multiple safety violations are recorded. Examples of these include deviation to the center lane (V_{DTL}), sharp braking (V_{BrD}), localization loss (V_{NDTLs}), collisions (V_{Col}), and violations of distance to collision (V_{DTC}VDTC).

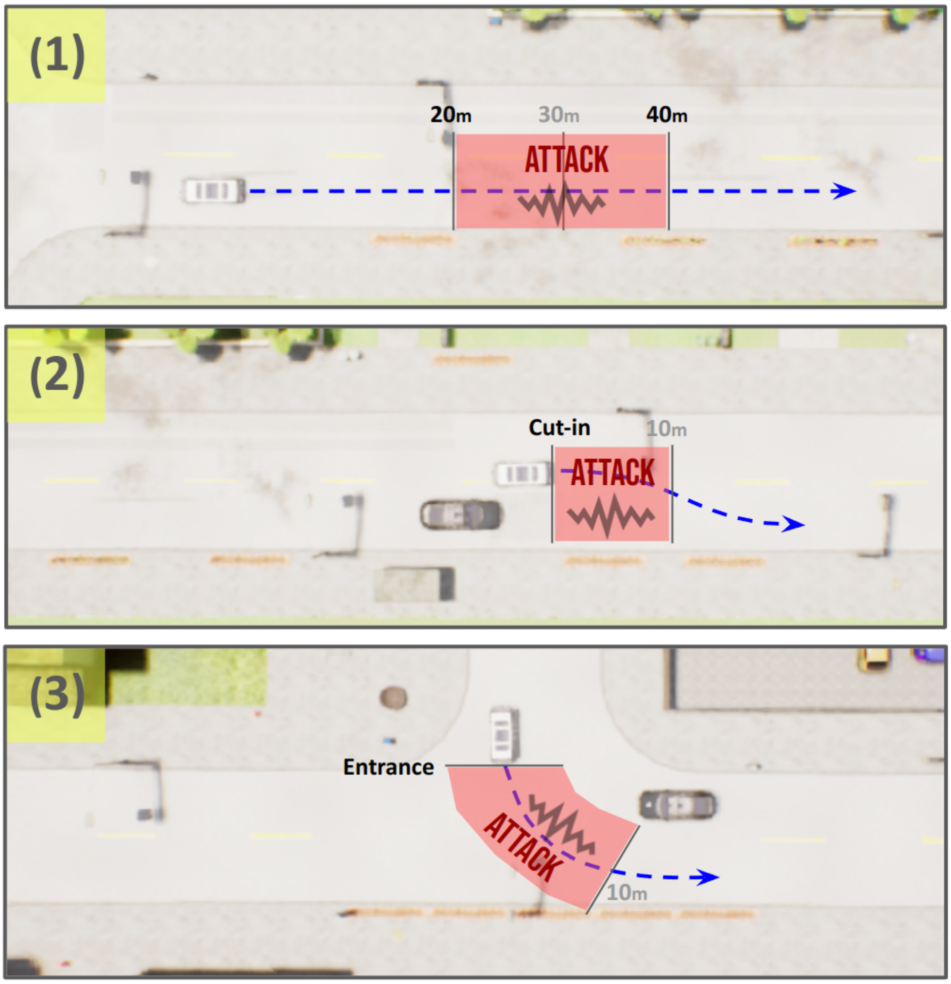


Figure 38: Game-engine view of three simulated scenarios representing the attack occurrence place during the mission; 1) Straight-line 2) Overtake 3) Intersection.

In these cases, while the AV may exhibit unsafe behaviors or suboptimal performance, it is still able to complete the mission.

Two critical safety metrics are sidewalk incursions (V_{SiIn}) and collisions (V_{Col}), both representing severe safety hazards. A sidewalk incursion indicates where the AV veered off its intended path and encroached into pedestrian zones, potentially endangering people on sidewalks. Similarly, a collision signifies an event where the AV collided with a nearby NPC vehicle.

Another key performance indicator is the deviation to the reference path (Dev2Ref), which measures how far the AV strayed from its intended trajectory. It is important to note that Dev2Ref is not the deviation at a single point; rather, it represents the summation of the deviations at several reference points along the planned path to the actual route traveled by the AV. This cumulative nature of the metric results in larger values, especially when the AV frequently deviates from the intended trajectory.

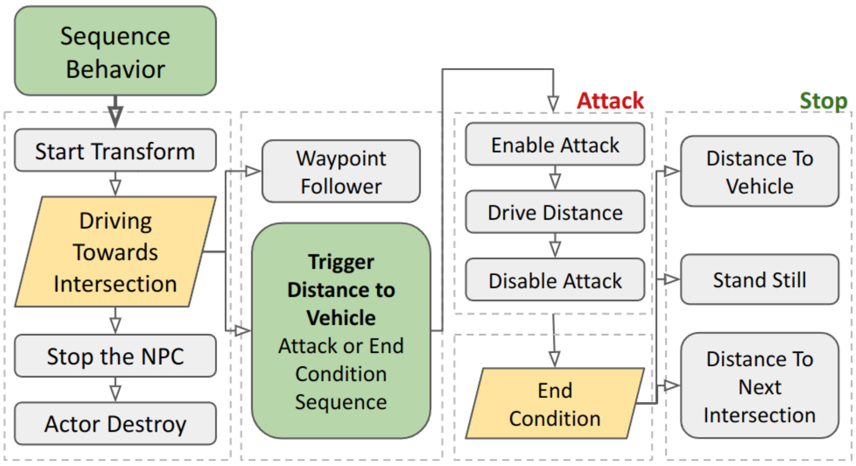


Figure 39: Flow-graph of how each scenario is processed in the simulation platform.

Table 25: Safety Evaluation Criteria

Safety Condition	**Data Label**	**Description**	**Metric**
Not Finished	*NotF*	Failure to finish the mission	Pass/Fail
Sidewalk Incursion	*SiIn*	AV deviation into pedestrian zone	Pass/Fail
Collision	*Col*	AV collides with NPC	Pass/Fail
Distance-to -Collision	*DTC*	Violation of the safe distance between AV and NPC	AV within 0.5m of other vehicle
Distance-to -Centre Lane	*DTL*	Violation of the safe distance between AV and Centre Lane	AV within 0.4m of centre lane
Break on Driving Lane	*BrD*	AV initiates emergency break on driving lane	Pass/Fail
Localization	*NDTLs*	Localization Loss	NDTerror $>$ 1.0
Violation	*V*	Safety Violation	

3.3.5 Results

For each of the scenario's, the results, as expressed in Tables. 27, 28, 29 demonstrate that increasing level of noise and duration of the EMI attack impact the safety and performance of the AV.

The manipulation of the steering sensor input, at higher noise levels, affects the feedback-loop for calculation of localisation which results in the AV experiencing loss and jumps of localisation. The NDT algorithm, used in the localisation algorithm, exhibits weakness in holding the position of the AV during sensor manipulation, which is demonstrated by loss of localisation, in attempting to re-correct, it incurs jumps. The loss and jumps of the localisation affect the displacement of the AV as such the cost-based algorithm used by the mission and motion planning module, recalculates the trajectories and chooses a new roll-out. The choice of new trajectory of the AV disrupts the flow of critical maneuvers within

Table 26: *Performance Evaluation Criteria*

Performance Criteria	**Data Label**	**Description**	**Metric**
Lane Transition	*RlOut*	AV executes multiple roll-out transition	Pass/Fail
Localization	*NDT*	Localization Performance	AV localization matching
Localization	*NDTer*	Mean localization pose error	Localization error margin
Duration	Dur	Duration in seconds	
Max NDT score	MxNDTSr	Max NDT score during a mission	Smaller = Better
Path Deviation	Dev2Ref	Sum of deviation to the reference path in sampled points	Smaller = Better
Max Lat Deviation	MxLaDev	Max lateral deviation from original path	Smaller = Better

the scenario such as the cut-in process of overtaking, smoothing of trajectory in keeping straight-line and turning at the intersection.

Scenario 1: Straight-Line

Within the Straight-Line Scenario Safety Results (Table. 27), safety violations begin to occur when 0.05 noise is introduced into the sensor input, marking the threshold where the AV system starts to struggle with maintaining safety. At this noise level, a 10% safety violation rate provided by lateral deviation violations was observed. As the noise level and attack duration increase, the AV experiences a progressive degradation in performance, culminating in the highest noise level (0.2) and the longest attack duration (20 meters), which results in a 42% safety violation rate and 38% lateral deviation violation.

A key characteristic of the AV's behavior in this scenario is the Deviation-to-Centre-Lane. The noise is injected into the steering sensor, and abrupt changes in the steering actuation cause the vehicle's control system to oscillate between making corrections and following the desired path. Autoware's motion planner attempts to rectify the vehicle's course, but the corrections are often sub-optimal, resulting in the AV veering to a dangerous proximity to the center line. This behavior indicates a weakness in the resilience of the AV's planning algorithm when recovering from anomalous inputs, as the system fails to regain optimal performance after the attack.

A more extreme example of dangerous trajectories, is where the EMI injection causes the AV to lose localisation which, cascades to affect the decision-making of the planning algorithm. The attack localization loss, as indicated by the NDT Error Value and NDT Score increasing, and the sharp variances between autoware and simulator. This behaviour results in the AV veering into the adjacent lane and hitting the side curb, a behaviour characteristic of 6% of the runs within the maximum noise and duration simulation set. Associated with these safety violations are significant performance degradation. In scenarios with low noise levels (0.01), the maximum lateral deviation is limited to around 0.2 meters. However, under maximum noise (0.2) and 20-meter duration conditions, the lateral deviation increases dramatically to 8.2 meters, showcasing the substantial impact of noise on the AV's ability to maintain its path. This severe lateral deviation illustrates the danger posed by noise-induced errors in the vehicle's steering and localization systems.

Table 27: Summary of the Safety and Performance Evaluation - Straight Line Scenario. The first line is our baseline path where no attack was applied.

SAFETY							
Length	Noise	NotF	SafetyV	V_{SiIn}	V_{DTL}	V_{NDTLs}	V_{BrD}
-	baseline	0%	0%	0%	0%	0%	0%
10 m	0.01	0%	0%	0%	0%	0%	0%
10 m	0.05	10%	10%	0%	10%	0%	0%
10 m	0.1	12%	12%	0%	6%	6%	0%
10 m	0.2	30%	30%	2%	26%	12%	8%
20 m	0.01	0%	0%	0%	0%	0%	0%
20 m	0.05	34%	34%	2%	30%	8%	2%
20 m	0.1	34%	36%	4%	28%	18%	6%
20 m	0.2	42%	42%	6%	38%	14%	2%

PERFORMANCE					
Length	Noise	Dur	RlOut	MxLaDev	MxNDTSr
-	baseline	57.6s	0	0.1m	11.9
10 m	0.01	59.9s	0	0.2m	11.9
10 m	0.05	61.5s	0.16	1.6m	12.0
10 m	0.1	65.5s	0.3	1.5m	12.5
10 m	0.2	71.8s	1.18	8.3m	25.5
20 m	0.01	70.2s	0	0.3m	14.2
20 m	0.05	75.6s	0.94	1.7m	25.5
20 m	0.1	82.6s	1.36	8.2m	46.9
20 m	0.2	85.6s	1.64	8.2m	35.0

Moreover, the *RlOut* metric—which tracks the average number of local trajectory transitions during a mission—shows a significant increase under high-noise conditions. This indicates the motion planner's growing uncertainty and inability to maintain a stable trajectory. As the AV continuously switches between trajectories, it struggles to converge on an optimal path, leading to erratic driving behavior and further deviations. Another factor exacerbating these challenges is the increased mission duration under noise attacks. The AV, displaced from its efficient path due to trajectory deviations and localization errors, takes longer to complete the mission. In the 0.2 noise / 20-meter scenario, the mission duration extended by nearly 28 seconds compared to the no-attack baseline, reflecting the inefficiency introduced by the noise attacks.

Scenario 2: Overtake Maneuver

In this experiment, the attack length was fixed at 10 meters while varying the noise levels to assess their impact on the vehicle's performance and safety. In the no-attack scenario (see Table. 27), the AV successfully completed the overtaking maneuver with minimal disruptions. The mission failure rate (NotF) was 0%, and a 1% violation of distance to collision (V_{DTC}) was recorded, indicating that in one case, the vehicle exceeded the safe distance from nearby objects. Despite this, there were no sidewalk incursions (V_{SiIn}), collisions

(V_{Col}), or localization loss (V_{NDTLs}). The vehicle maintained a safe average DTC of 0.4 meters. The mission duration was 104.7 seconds, with an NDT error of 0.2 and a standard deviation of 0.1.

Table 28: Summary of the Safety and Performance Evaluation - Overtake Scenario. No attack was carried out in the baseline experiment.

SAFETY							
Noise	NotF	SafetyV	V_{SiIn}	V_{Col}	V_{NDTLs}	V_{DTC}	V_{BrD}
baseline	0%	1%	0%	0%	0%	1%	0%
0.01	7%	18%	2%	3%	4%	14%	1%
0.05	16%	23%	8%	3%	11%	10%	2%
0.1	29%	40%	18%	2%	26%	14%	1%
0.2	33%	39%	23%	7%	28%	14%	2%

PERFORMANCE						
Noise	Dur	RlOut	$\overline{DTC}$	MxNDTSr	NDTer	S-NDTer
baseline	104.7s	8.2	0.4m	19.4	0.2m	0.1m
0.01	107.3s	7.8	0.2m	55.9	0.2m	0.2m
0.05	121.4s	8.9	0.2m	73.9	0.4m	0.5m
0.1	125.4s	10.0	0.2m	63.9	0.7m	0.9m
0.2	124.7s	10.2	0.2m	53.1	0.6m	0.8m

In the 0.01 noise scenario, V_{NotF} increased to 7%, and by the 0.2 noise level, it reached 33%. Similarly, $V_{NDTLoss}$ was first observed at 0.01 noise (4%), growing to 28% in the 0.2 noise scenario. These results indicate that noise in the sensor input significantly disrupts the vehicle's ability to maintain accurate localization, directly impacting mission success.

In the no-attack scenario, V_{SiIn} and V_{Col} were recorded at 0%, reflecting ideal behavior where the AV stayed within its designated path and successfully avoided NPCs during overtaking. However, as noise levels increased, both metrics worsened. In the 0.01 noise scenario, V_{SiIn} rose to 2%, and V_{Col} to 3%, showing the system's diminished capacity to maintain lane discipline and avoid nearby vehicles. At the highest noise level (0.2), sidewalk incursions increased to 23%, while collisions reached 7%, a significant rise indicating the AV's inability to safely manage the overtaking maneuver under heavy noise interference. These results suggest that sensor noise not only disrupts the vehicle's path but also critically impacts its ability to avoid hazards that could lead to severe accidents involving both pedestrians and other vehicles.

The V_{DTC}, which reflects the rate at which the AV exceeded safe distances from nearby objects, increased from 1% in the no-attack case to 14% in the 0.2 noise scenario. This was accompanied by a rise in sharp braking events as the AV's control system struggled to compensate for the noisy input, leading to more frequent sudden stops. As the noise level increased, the RollOut metric showed greater instability. In the 0.2 noise case, the RollOut metric increased from 8.2 (in the no-attack scenario) to 10.2, indicating the planner's increasing uncertainty in maintaining a stable trajectory.

The mission duration increased as the noise level rose. In the 0.2 noise scenario, the AV took 124.7 seconds to complete the maneuver, an increase from 104.7 seconds in the no-attack scenario. Additionally, the NDT error and its standard deviation saw significant increases, with the NDTer rising from 0.2 to 0.6 and the S-NDTer increasing from 0.1 to 0.8, highlighting the degradation in localization performance under noisy conditions.

Scenario 3: Intersection

In the intersection scenario, the attack length remained unchanged at 10 m, while the noise levels varied to assess their impact on the AV's performance during this complex maneuver. In the baseline scenario, the AV successfully navigated the intersection without mission failure (0%) or significant safety violations, aside from a small 3% V_{DTC}. There were no recorded V_{SiIn} or V_{Col}, and the AV maintained an average DTC of 0.4 meters, with an NDTer of 0.1 and a minimal deviation from the reference path of 20.4 meters. The overall mission duration was 65.8 seconds, and the system performed with only 2.2 RollOut changes, indicating a stable and efficient planning process.

Table 29: Summary of the Safety and Performance Evaluation - Intersection Scenario. No attack was carried out in the baseline experiment.

SAFETY							
Noise	NotF	SafetyV	V_{SiIn}	V_{Col}	V_{NDTLoss}	V_{DTC}	$\overline{DTC}$
baseline	0%	3%	0%	0%	0%	3%	0.4m
0.01	8%	15%	0%	1%	7%	10%	0.2m
0.05	19%	27%	2%	3%	16%	13%	0.2m
0.1	23%	32%	6%	3%	19%	16%	0.2m
0.2	25%	28%	4%	4%	22%	7%	0.1m

PERFORMANCE							
Noise	Dur	RlOut	MxNDTSr	NDTer	S-NDTer	Dev2Ref	S-Dev2Ref
baseline	65.8s	2.2	38.5	0.1m	0.1m	20.4m	8.2m
0.01	70.5s	3.1	39.5	0.2m	0.2m	39.1m	98.8m
0.05	72.9s	3.9	40.9	0.4m	0.4m	63.6m	170.4m
0.1	74.2s	4.5	37.5	0.5m	0.5m	69.6m	147.5m
0.2	74.5s	4.1	39.1	0.4m	0.5m	77.9m	154.9m

As noise levels increased, the NotF rate rose from 8% at 0.01 noise to 25% at 0.2 noise. Safety violations also saw a sharp increase, particularly in terms of V_{NDTLs}, which jumped from 7% at 0.01 noise to 22% at 0.2 noise. This degradation in localization directly impacted the AV's ability to make timely decisions and follow the intended trajectory, leading to more dangerous driving behavior.

While sidewalk incursions and collisions were rare in the baseline scenario, they became more frequent as noise levels rose. At 0.2 noise, 4% of the runs resulted in V_{SiIn}, and 4% in V_{Col} with NPCs within the intersection. This behavior indicates a critical safety failure, where the AV not only lost control of its lane discipline but also failed to avoid NPCs and pedestrian zones.

The cumulative deviation remained relatively low in the no-attack baseline scenario, indicating stable performance. However, under the influence of noise, this deviation increased significantly. For example, in the 0.2 noise scenario, the Dev2Ref reached 77.9 meters, with a high standard deviation of 154.9 meters, demonstrating the system's growing instability under attack. The high standard deviation reflects the inconsistency in the

AV's ability to maintain a predictable trajectory, as deviations varied considerably at different points along the path. The increasing Dev2Ref values show that the AV struggled to recover from noise-induced errors, leading to significant drift from the planned path.

The results show that the roll-out metric increased as noise levels rose. In the 0.01 noise scenario, the roll-out increased to 3.1, and by 0.2 noise, it rose to 4.1, indicating the planning system's growing uncertainty in selecting and maintaining a stable path. The maximum NDT score also fluctuated, reaching a high of 40.9 in the 0.05 noise scenario, highlighting the deteriorating localization performance.

The NDT error and its standard deviation also increased with higher noise levels. At 0.2 noise, the NDT error rose to 0.4, with a standard deviation of 0.5, indicating significant localization drift. This localization instability contributed to unsafe driving behavior, as reflected in the increased V_{DTC} and collisions. The mission duration also increased with noise levels, from 65.8 seconds in the baseline scenario to 74.5 seconds at 0.2 noise. This duration increase indicates the AV's struggle to efficiently navigate the intersection under attack, as the planning algorithm and control systems were frequently forced to adjust to counteract the noise-induced deviations.

Comparison Between Safety Violations and Simulated Scenario

Figure 40 represents radar graphs that provide a clear visual representation of the impact of noise attacks on the AV across all different mission types: straight-line driving, overtaking, and intersection maneuvers, with varying attack lengths (10 meters and 20 meters) for the straight-line scenario. By comparing these radar graphs, we can discern how the attack influences the AV in different maneuvers and understand whether the vulnerability is related to the nature of each maneuver.

In the straight-line scenario (Figure 40 (a) and (b)), the radar plots show a clear difference between the 10-meter and 20-meter attack lengths. With the 10-meter attack (Figure (a)), the V_{DTL} and V_{NDTLs} are relatively contained at noise levels below 0.1, but they spike at 0.2 noise, indicating that longer attack lengths exacerbate the vehicle's struggle to maintain its trajectory. By contrast, in the 20-meter attack scenario (Figure (b)), the impact of noise is more pronounced across all noise levels, with a higher percentage of NotF and significantly greater V_{DTL} and V_{NDTLs} values. This suggests that the longer attack duration amplifies the system's inability to recover from perturbations in the steering sensor, causing the AV to deviate further from the planned path.

In the overtaking scenario (Fig. 40 (c)), the radar plot highlights that this maneuver is particularly vulnerable to V_{NDTLs} and V_{DTC} as noise levels increase. Even at 0.01 noise, the AV shows a marked increase in these safety violations, and by 0.2 noise, V_{NDTLs} and V_{DTC} reach critical levels. This indicates that overtaking is a more complex and challenging maneuver for the AV compared to straight-line driving, as it requires the vehicle to safely execute lane changes and avoid collisions with NPCs. The complexity of coordinating between localization, path planning, and collision avoidance makes the system more prone to safety violations when noise is introduced.

In the intersection scenario (Fig. 40 (d)), the radar plot demonstrates that this maneuver is less affected by V_{DTC} compared to the overtaking scenario, but the mission failure rate and localization loss are notably higher. Even at 0.01 noise, NotF jumps to 8%, and V_{NDTLs} reaches 7%, while at 0.2 noise, NotF reaches 25%, indicating a substantial failure rate. The intersection maneuver places a high demand on the AV's localization and planning systems, as it requires precise decision-making in a constrained environment with multiple potential collision points. The increase in safety violations with rising noise levels reflects the difficulty the AV faces in maintaining control during complex navigation

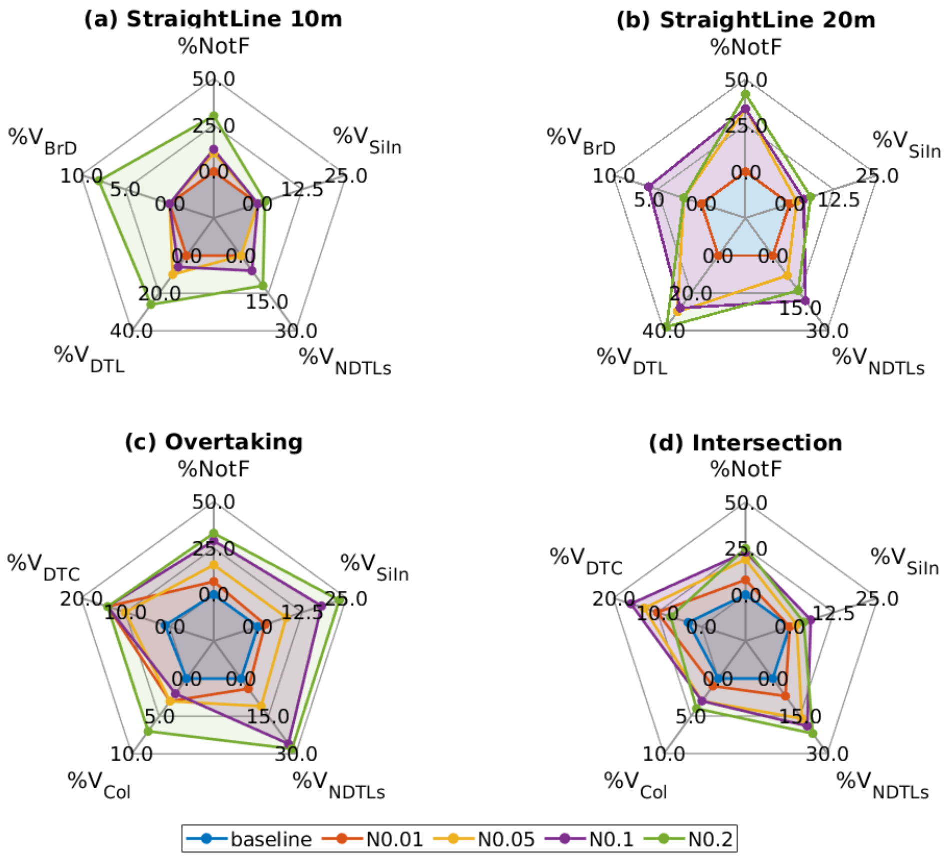


Figure 40: Safety violation of simulated scenarios.

tasks in intersections, where it must simultaneously monitor multiple potential threats and adjust its trajectory.

The vulnerability of the AV to noise attacks appears closely tied to the nature of the maneuver. Straight-line driving is less demanding in terms of control and localization, and as a result, the AV is able to handle noise better—though longer attack durations (as in Fig. 40 (b)) significantly increase the risk of mission failure. In contrast, overtaking involves more dynamic path changes and collision avoidance, making it more susceptible to noise, as seen in the sharp rise in V_{DTC} and V_{NDTLs} even at low noise levels. Intersection maneuvers also present significant challenges, particularly due to the need for precise localization and decision-making at multiple points, resulting in higher mission failure rates and localization loss as noise levels increase. These findings suggest that the more complex the maneuver (i.e., those requiring more dynamic control and interaction with external factors like NPCs or intersection points), the more vulnerable the AV is to noise attacks.

Violation to noise correlation analysis

The correlation heatmap shown in Figure 41 reveals significant insights into how different safety violations and performance metrics are affected by noise levels across various maneuvers and attack durations. Among all the maneuvers, straight-line driving (10m attack) demonstrates the highest correlation between noise levels and mission failure, with a co-

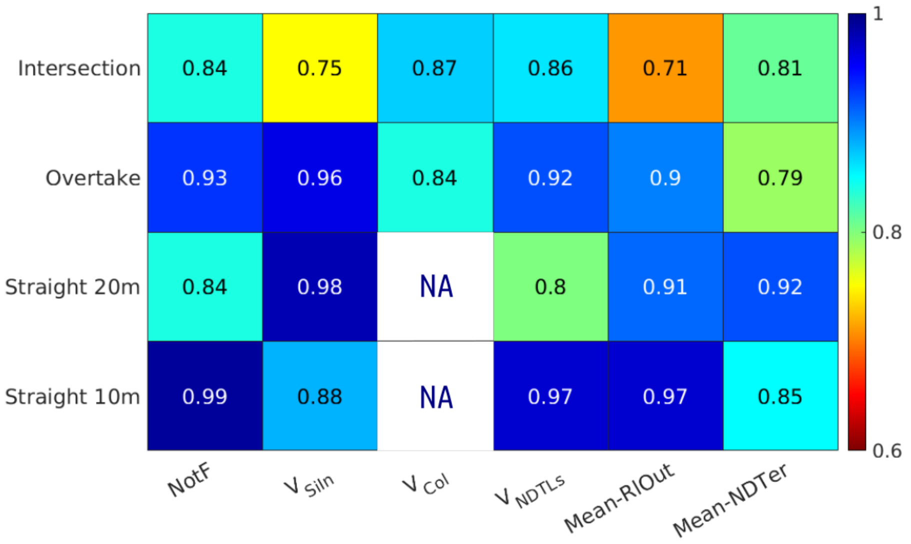


Figure 41: Correlation coefficients between violation metrics (horizontal axis) and noise levels ([0, 0.01, 0.05, 0.1, 0.2]) for each scenario (vertical axis). The values indicate the strength of the relationship between the likelihood of each violation and changes in noise levels.

efficient of 0.99, indicating that shorter attack duration in straight-line driving are highly sensitive to noise. The overtake scenario follows this with a correlation of 0.93. Both the intersection and straight-line 20m scenarios show a correlation of 0.84 for mission failure, suggesting that longer attack duration and intersection maneuvers are somewhat less sensitive to noise, possibly due to the nature of the mission. Regarding sidewalk incursions, longer attack duration in the straight-line (20m) and overtake scenarios show the strongest correlations, at 0.98 and 0.96, respectively. In contrast, the intersection maneuver displays the weakest correlation for sidewalk incursions, reflecting the controlled, slower nature of this maneuver.

When examining localization loss, straight-line 10m and overtake show the highest correlations, 0.97 and 0.92, respectively, indicating that these scenarios are most affected by noise in terms of localization. The intersection scenario, though still sensitive to noise (0.86), shows a somewhat lower correlation, likely due to the AV's reduced speed and static behavior at stop points. Collision, on the other hand, shows similarly strong correlations in overtaking (0.84) and intersection (0.87) scenarios, but this metric is irrelevant in straight-line driving, as there are no NPCs involved in those maneuvers. The correlation for RollOut switches is also highest in straight-line 10m attacks (0.97), followed by straight-line 20m and overtake, while intersections have the lowest correlation (0.71) in this category. For NDTer, longer attack durations in straight-line scenarios show the highest correlation (0.92), while intersections and overtakes show lower values.

Overall, the straight-line (10m) and overtake scenarios exhibit the highest sensitivity to noise across several metrics, such as mission failure, sidewalk incursions, and localization loss. Intersection scenarios, in contrast, show consistently lower correlations, likely due to the nature of the maneuver, where the vehicle slows down or stops, reducing the dynamic impact of noise during attacks. This behavior at intersections explains the weaker overall correlation with noise, as the AV is generally at lower speeds and is less engaged in continuous movement compared to the overtake and straight-line scenarios. This highlights how the nature of each maneuver, particularly its dynamic or static characteristics, influences the vehicle's vulnerability to noise-induced safety violations and performance degradation.

3.3.6 Discussion

Throughout the paper, we demonstrated that AD software is sensitive to EMI attacks that can generate different levels of safety violations from low-priority violations, from which the vehicle can recover but resulting in suboptimal behaviour, to severe violations causing collisions or endangering other road users.

RQ1 How does a manipulation to the electromechanical component propagate through the AD software stack?

From our results, it emerges that an EMI attack at the steering sensor level often causes SiIn, DTL, or DTC violations, which are the most commonly visible in Figure 40. To back-step this behaviour, to eventually debug such a complex AD software stack in a general purpose approach, developers will require an accurate analysis of each block in terms of data input-output relation. In our case, we carried out a back-step analysis at the ROS-topic level to identify the nodes that subscribe to specific messages. Here, we found out that the most probable user of steering sensor data, thus generating violations, is the mission and motion planning module, visible in Figure 12, and composed of several sub-blocks including `op_trajectory_generator` and `op_waypoint_follower`, that represent the most probable components generating wrong decisions. While at the low level, PID controllers might be able to withstand noise to some extent, intelligent controllers have shown inherent vulnerability to this attack propagating from the low level up as raw sensor data to the master controller and up to the ROS topics.

RQ2 What dependencies exist between the AD control algorithm and low-level control?

High-level intelligent controllers trust digital data flowing over the in-vehicle network communication level. The interdependence of control algorithms resides in the feedback loop reading data from the low level while the AD acts in a hybrid deliberate/reactive robotic paradigm. In such a paradigm, well studied in robotics, an AD reacts quickly upon sensing without performing global-planning, which is typically a computationally demanding task running concurrently. SiIn, DTL, or DTC violations, which are the most commonly found in our analysis, are a typical result of the reactive behaviour of ADs. Similarly, the planner might generate unsafe trajectories in case of localization data corruption such as NDTLs violation or increase in NDTer margin. Eventually, the vehicle can recover from some violation when the global-planner generates a new waypoint, but this is not always guaranteed when some stochasticity is involved in the process.

RQ3 Where in the architecture of the autonomous vehicle can defensive mechanisms be placed to defend against control invariants?

Strategies to detect and mitigate low-level sensor data input manipulation focus on redundancy and multiple levels of data integrity checks. To investigate this question we step through each of the layers of the AV:

- **Low-Level PID Controller:** Integrity and plausibility checking of the PID can mitigate but not stop the injection of anomalous sensor input values. The PID has its own robustness, which is mathematically proved, the PID lacks the intelligence to interpret the meaning behind the input data. Therefore, attacks which manipulate the sensor input always have the possibility of traversing the PID. It is also possible to implement analog filters and hardware saturation, however, as mentioned, at this level, there is no means to discern attack behaviour which resembles regular signal/circuit specification and its operating characteristic.

- **Intermediate Layer:** At this level, it is possible to conduct inspection of the CAN data. The master controller has low-computational capacity. Therefore, implementation of mechanisms to interpret and provide intelligence of the CAN data is limited. Data saturation and filtering is possible at this level. However, filtering and saturation strategies would be challenged to defend against an adaptive sensor manipulation attack which searches for the filtering and saturation parameters and develop a 1-step or n-step attack which falls outside the range.

- **High-Level Control Layer:** A redundant, fall-back controller has a cost in terms of financial, compute and network resources, and cannot guarantee that an attack would also aim to manipulate the redundant controller. Furthermore, redundant controllers accessing the same sensor data might generate the same unexpected behaviour.

Our recommendations, for this particular use case, is to accurately model the sensor behaviour at the physical level considering the physical world world we live in. In this context, sensors, such as everything else, should obey Newton (for motion) and Maxwell equations (for electromagnetism). To detect sensor data anomaly our knowledge of the physical model of the sensor can be utilised to predict variances to this model. This would effectively detect a possible attack much earlier and thus prevent DTC & DTL violations occurring in the motion planning block. The validation of sensor data can run in a concurrent process throwing exceptions in case of unexpected levels of noise. The response action to an exception need to be modelled on the level of risk.

3.4 Summary

Within this section we developed methods for cybersecurity testing of AD software and illustrated their utility for vulnerability discovery by conducting applied, experimental testing. Precise metrics that incorporate safety, which represent the integration and stability of vehicle dynamics and autonomous software control, and security, which represent the parameters of the attack model, enable the ability the discern the affect of cyber attacks to the semantic properties of AD software. Malicious injection and time-delay attacks targeted at the perception and planning modules, and the low-level actuation sensing, results in malicious input propagating through the software to affect the reliability and safety of control decisions. From the conducted sensitivity analysis, vulnerabilities of the software modules can be characterised as a lack of robustness to malicious injection of input data at parameter ranges which represent minimal deviation. Whilst the sensitivity ranges present a finding in terms of the case study vehicle, iseAuto, these values will differ based on the class of vehicle (light-passenger, heavy rigid) and the design of the control system. Therefore, the novelty lies in the overarching methods used to distinct the impact of cyber attacks to the software and vehicle dynamics and testing approach used to deliver the attack and generate feedback of the system. Furthermore, the results, within the context of applied, experimental testing on a real-world system, illuminates the gap in comprehensive scenario-based testing where cyber attack test cases are considered. It further highlights the need for integration of control software design processes and test feedback. The next section contends with this issue through investigating techniques to assist software and control system designers with debugging and root-cause analysis.

4 Debugging Autonomous Control Software to Cyber Attacks

4.1 ADAssure: A Debugging Methodology for AD Control Algorithms

AVs are increasingly being utilised for transportation on public roads. Waymo and Cruise offer AD ride-hailing services in San Francisco, Apollo Baidu in China, and numerous such services are operating in Europe. Central to the wider-adoption of AD vehicles on public roads is the security and safety of their control algorithms that enable self-driving technology. AD control algorithms comprise a complex code-base of interconnected modules that perform tasks and sub-tasks that enable a vehicle to sense, perceive, localise, and navigate in a driving environment. With the increase in diversity of AD use-cases from valet parking to public transportation in public traffic, the code base of AD control algorithms will reputedly grow from 100-200 million to billions of lines of code [28].

Within this complex environment, debugging the code for logical errors arising from unexpected control behaviour is a fundamental challenge [330]. AD system designers need to pinpoint where in the control software weaknesses are, in order to focus debugging efforts in an efficient manner. Existing studies attempt to rectify unexpected AD control behaviour at run-time through smoothing trajectories utilising neural networks [41] [137] [173]. The applicability of these studies in real-world AD programs are limited due to the highly dynamic environment of autonomous driving and the probabilistic nature of the algorithms for planning.

Furthermore, in these studies, the analysis lacks the expertise from the algorithm designer and safety engineer to inform on the nature of the behaviour of vehicle dynamics, whether noise identified as irregular could be considered for a control engineer within normal constraints, whether AD behaviour could be considered a legitimate safety response to an unexpected event and whether the parameters for which the run-time solution is designed are appropriate for differing class of vehicles with different dynamic profiles. We consider the design phase to offer the most promising area of initial investigation to improve the robustness of control algorithms, which can be translated to real-world AD systems.

We propose ADAssure, a methodology for debugging control algorithms during the design-time phase of AD control software development (Figure 42). ADAssure is built upon the idea that the data of vehicle dynamics and sensing of AD systems can be analysed for anomalous control behaviour, which can then be transformed into assertions on the AD control. We use association rules that enable us to mine datasets of varying scales and fingerprint the pattern of anomalous activity. These rules can be used to guide AD system designers to focus on the debugging of the control algorithms. To evaluate ADAssure, we focus on a control system algorithm used in a real-world AD vehicular system providing ride-hailing services.

4.1.1 ADAssure: Methodology

The development of ADAssure has three main motivations. First, it aims to provide AD system designers with a methodology to identify and fix vulnerabilities that align with the design of AD algorithms. Second, given the ever-changing nature of the autonomous vehicle system, it strives to establish a structured methodology that allows for consistent, flexible, and repeatable testing. Third, it aims to support unit testing, allowing testing of individual components of the autonomous system in isolation from other dynamic factors affecting autonomous control.

The foundations of the ADAssure methodology are based on the analysis of the vehicle dynamics and sensing data to guide the creation of assertions of the vulnerability of

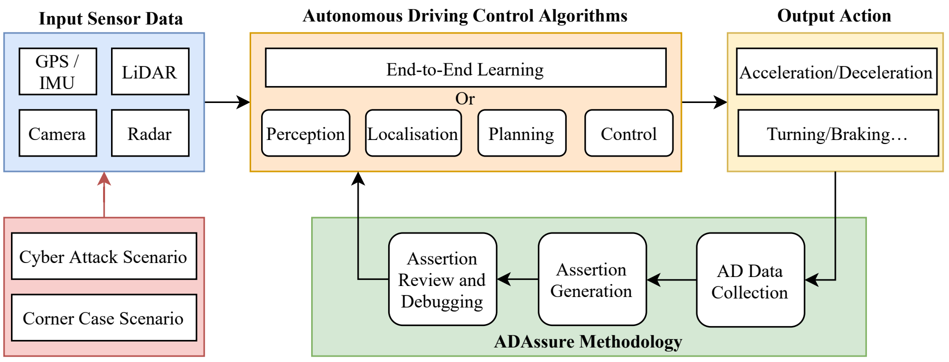


Figure 42: Comprehensive ADAssure methodology overview that illustrates each step of the process, from data collection to assertion creation, review of assertions, and debugging.

the AD control algorithms. The analysis consists of a sensitivity analysis of vehicle dynamics data (e.g., velocity, yaw, and steering angle), sensor data (e.g., lateral and longitudinal movement), and visualisation of the trajectory of the AD system. This helps identify key parameters to build assertions of the AD control algorithms. The AD control system designers can use the assertions to identify and locate the vulnerabilities of the control model and develop mechanisms to test and fix the errors. The ADAssure methodology comprises three main phases: AD Data Collection, Association Rule Generation, and Assertion Review and Debugging. Next, we will explore each phase in more depth.

Autonomous Driving Data Collection

This phase consists of generating data from the real-world system or simulation environment. The benefit of a simulation environment is that driving scenarios can be automated or designed to test a specific condition, such as a cyber-attack or a corner case. The data output is structured according to established metrics. These can be vehicle dynamics parameters (yaw angle, velocity, etc.), sensing data (position co-variance, point-cloud, etc.), and safety parameters (distance-to-collision, etc.). The AD data is outputted in a format that can be interpreted by analytical tools, in our use-case, .csv format.

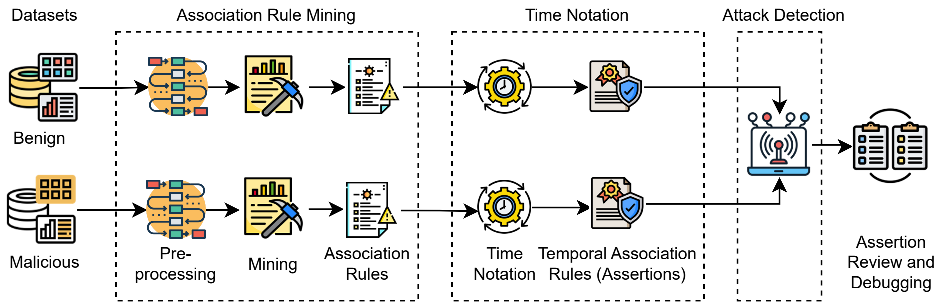


Figure 43: Phases for Assertion Generation

Algorithm 1 Association rule mining & time notation

```
Input: $\mathscr{N}$, $\mathscr{D}$   Output: $next[\mathscr{N}] = antecedent \rightarrow next[\mathscr{N}]consequent, before[\mathscr{N}] = antecedent \rightarrow before[\mathscr{N}]consequent$   {*}[l]Initialization and Preprocessing $\mathscr{R}$ = $antecedent \rightarrow consequent$
for all $f \in \mathscr{D}$ do $\mathscr{D}'$ = $\mathbf{MoveUp}(f(\mathscr{N}))$   {*}[l]Mining $\mathscr{R} \leftarrow \mathbf{apriori}(\mathscr{D}')$   {*}[l]Time Notation
    if ($\mathscr{R}.antecedent$ == ($t \in \mathscr{D}'$)) and ($\mathscr{R}.consequent$ == ($f \in \mathscr{D}'$)) then $next[\mathscr{N}] \leftarrow$ label($\mathscr{R}$)
        if ($\mathscr{R}.antecedent$ == ($f \in \mathscr{D}'$)) and ($\mathscr{R}.consequent$ == ($t \in \mathscr{D}'$)) then $before[\mathscr{N}] \leftarrow$ label($\mathscr{R}$)
```

4.1.2 Association Rule Generation Phase

The goal of this phase is to process the data generated from the previous phase and produce a set of association rules that can be translated into assertions in the Assertion Review and Debugging phase. This phase is comprised of three primary steps (as shown in Figure 43):

1. Association Rule Mining,
2. Time Notation,
3. Attack Detection.

The association rule mining is applied to both benign and malicious datasets, resulting in two distinct sets of association rules. These rules are then processed through the Time Notation step to incorporate temporal information, yielding temporal association rules (assertions) in the form of $next[\mathscr{N}]$ and $before[\mathscr{N}]$ patterns. We define $next[\mathscr{N}]$ type of rule in the general form of $\mathscr{X} \rightarrow next[\mathscr{N}]\mathscr{Y}$. This rule indicates that when $\mathscr{X}$ occurs, after $\mathscr{N}$ time instants, $\mathscr{Y}$ will occur. $\mathscr{N}$ is a positive integer value. Moreover, we define $before[\mathscr{N}]$ rule in the general form of $\mathscr{X} \rightarrow before[\mathscr{N}]\mathscr{Y}$. This rule demonstrates that whenever $\mathscr{X}$ happens, $\mathscr{Y}$ should have occurred $\mathscr{N}$ time instants before that. The "Attack Detection" step compares these temporal association rules, ultimately detecting attacks and anomalies within the datasets. Subsequent sections provide a more in-depth discussion of each step.

4.1.3 Association Rule Mining

This step primarily serves two objectives: pre-processing the datasets and subsequently mining association rules from the preprocessed data. To mine the association rules, apriori algorithm [102] was adopted and enhanced to mine temporal rules capable of detecting attacks at various time instances during autonomous vehicle (AV) operation. Algorithm 1 presents the details of the Association Rule Mining and Time Notation steps. In this algorithm, $\mathscr{D}$ denotes the dataset and $\mathscr{D}'$ is the preprocessed dataset, while f and t represent the dataset's features and target values. To prepare the dataset for mining the $next[\mathscr{N}]$ and $before[\mathscr{N}]$ temporal patterns, all the features of the dataset are moved $\mathscr{N}$ records above its original position. However, the target of the dataset remains as it is. Afterwards, the apriori algorithm is applied to the preprocessed dataset to mine a set of association rules. The output of this phase is a set of association rules in the general form of $antecedent \rightarrow consequent$ that are ready to be forwarded to the Time Notation step.

4.1.3.1 Time Notation In this step, the method integrates the concept of time into the association rules generated in the association rule mining step, leading to a set of temporal association rules. The method determines to which temporal pattern ($next[\mathscr{N}]$ or $before[\mathscr{N}]$) each extracted rule belongs and subsequently assigns the corresponding time label to the rule. If the antecedent value matches a target value in the dataset, and the consequent value has already been moved to another record in the dataset, the rule is labelled as a $next$ temporal association rule. Otherwise, if the antecedent of a rule mined in the association rule mining step matches a dataset feature that has already been moved to another record and the consequent of the rule matches the target value of the dataset, we label this rule as a $before$ temporal association rule. The mined rules are in the forms of $antecedent \rightarrow next[\mathcal{N}]consequent$, and $antecedent \rightarrow before[\mathcal{N}]consequent$, serving as assertions for debugging the AD system.

4.1.3.2 Attack Detection This step aims to identify rules indicating attacks on the AV. We assume that the sets of mined rules from the benign and malicious datasets should be similar under normal conditions, without any AV attacks. Any deviation between these rule sets signifies an anomaly in the autonomous vehicle. Per this assumption, the temporal association rules (assertions) mined during the time notation phase are classified into two sets. The first category comprises rules exclusively mined from the malicious dataset, lacking counterparts in the benign dataset. Any rule extracted solely from the malicious dataset, without a corresponding counterpart in the benign dataset, signifies an attack. These rules reveal abnormal behaviour in the malicious dataset, contrasting with different behaviour observed in the corresponding time instance of the benign dataset. Consequently, we classify these as attacks. The second category comprises similar rules mined from both benign and malicious datasets, but with different *minimum support* (`min_supp`) and *minimum confidence* (`min_conf`) values. The variations in these values indicate that, while the mined rules are similar, abnormal behaviours and anomalies exist between the datasets. The apriori algorithm employs these two metrics (i.e., `min_supp` and `min_conf`). The `min_supp` value is the threshold and a minimum value that is chosen by the expert to decide whether a rule occurs frequently in the dataset or not [107, 328]. The `min_conf` is the minimum value that is chosen by the expert and is an indication of how often a rule has been found to be true [102, 260]. Increasing the `min_supp` value results in fewer association rules that describe more general behaviour of the autonomous vehicle, while decreasing the `min_supp` value leads to rules covering rare behaviours (corner cases). Similarly, raising the `min_conf` value produces fewer but more valid rules. Valid rules refer to association rules that will not be violated with different attack scenarios like corner cases. These values in the ADAssure facilitate an effective attack detection process. The second category of rules aids the ADAssure in effectively identifying corner cases and the attacks that rarely occur on the AV. These rare attacks exhibit behaviour very similar to normal vehicle operation but are malicious and can lead to AV failure.

4.1.4 Assertion Review and Debugging

Within this phase, the association rules generated from the association rule mining are reviewed in conjunction with an analysis of the control behaviour and individual data parameters to develop assertions. Trajectory maps of the AD system and graphs, which demonstrate the sensitivity of the data parameters during benign and cyber-attack scenarios, are compared to the anomalous behavioral patterns detected by the association rule mining tool. Using expertise from the algorithm designer and safety validation engineer assists in understanding which parameters can uniquely demonstrate a vulnerability of an algorithm within the system. From developing an assertion on the system's vulnerability, the debugging effort focuses on a control flow analysis. As the assertion assists in pinpointing the specific module, the static analysis can focus on the control flow of the substituent functions within the module. As an example of the importance of this pinpointing, a local-planning module could have 15 diverse algorithms, and within these, each could have multiple different methods or functions. As the code of AD algorithms are differential equations, debugging can suggest optimisations that enable mitigation mechanisms against the identified vulnerabilities.

4.1.5 Autonomous Driving Control Algorithm

To evaluate the methodology, we focus on an AD control algorithm used in a real-world AD ride-hailing service. Within the AD pipeline, there are four key modules: localisation, perception, planning, and control. Within our study, we focus on the localisation and planning modules.

4.1.6 Experimentation and Results

To evaluate the impact of corner cases on AD system behaviour using the ADAssure methodology, we use datasets of corner cases from simulation and real-world driving from the target AD system. The 1st corner case scenario dataset is of three diverse cyber-security attacks on the AD system conducted in a simulation environment. As our focus is the planning and localisation algorithms, we used a low-fidelity simulation provided by *Autoware.AI* and the *OpenPlanner 2.5* planning algorithm. The 2nd corner case scenario dataset is of a Global Positioning System (GPS) spoofing event that occurred on the AD system during its operation on the roads of a capital city.

4.1.7 AD Control System Datasets

Cybersecurity Corner Case Dataset: Within this dataset, three attacks were conducted on the target AD vehicular system, which is attempting an overtaking maneuver. The three attacks are classified as:

1. *Lateral Position Offset Attack*
2. *Longitudinal Position Offset Attack*
3. *Message Time-Delay*.

In the lateral and longitudinal position offset attack, an attacker injects malicious data input into the lateral or longitudinal pose whilst the AD vehicular system is in the process of the overtaking manoeuvre (Figure 44). This attack could be conducted through GPS spoofing or interception and manipulation of the localisation sensor data. The attacker introduces a delay into the `current_pose` (lateral and longitudinal) sensor messages reaching the AD control pipeline for the message time-delay. The malicious data is injected at around the $21\,\mathrm{m}$ mark of the AV journey (travelled distanced) to the $67\,\mathrm{m}$. Each attack was conducted 300 times, accommodating a variation of different attack parameters. The lateral and longitudinal attacks introduced a deviation ranging from $0.16\,\%$ to $1.0\,\%$, which equates to around $20\,\mathrm{cm}$ to $1\,\mathrm{m}$. The message time-delay introduced delays of $0.3\,\%$, $0.6\,\%$, $1.0\,\%$ second, as a message is transmitted every $20\,\mathrm{ms}$, this range represents a delay of 15 to $50\,\mathrm{messages}$. In total, the dataset comprises over $1500\,\mathrm{scenario}$ runs of attacks and benign safety cases.

GPS Spoofing Real-World AV Dataset: The AD ride-hailing service transmits its sensor data via a logging node to an edge server, which stores the AD System data in a database. During its operations near the port area of the city, the AD vehicle encountered a loss of localisation from a GPS spoofing event which also affected other GPS-enabled platforms. This GPS spoofing continued intermittently throughout the preceding months. The

Table 30: AD System Data.

AD Data Type	Description
AV_X	Longitudinal Position of the AD System as to the HD Map
AV_Y	Lateral Position of the AD System as to the HD Map
AV_Steer	Steering Angle of the AD System
AV_Vel	Velocity of the AD System
AV_Yaw	Orientation of the AD System based on its centre of gravity
Roll-out_Num	Current Lane according to the lane selector of the AD Control Algorithm
DTC	Distance to collision of the AD vehicular system to the overtaking vehicle.
Position Co-variance	GPS position co-variance
Altitude	Altitude derived from the GPS

dataset used in this study is from the logging system of AD ride-hailing service.

AD System Data: The simulation and real-world datasets were structured to output data as shown in Table 30.

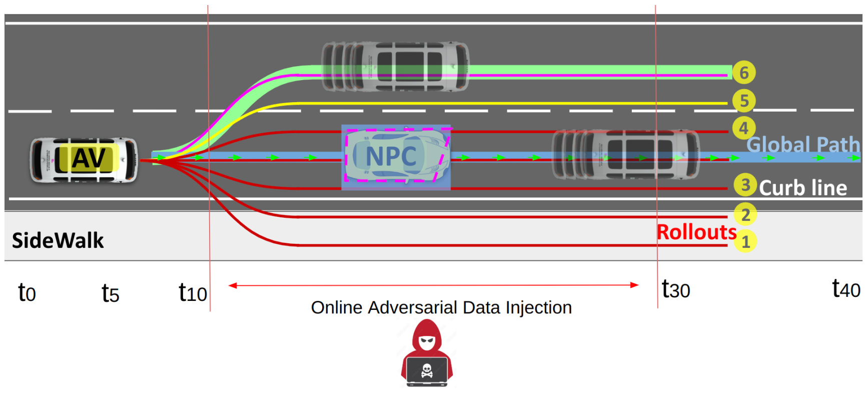


Figure 44: The threat model used for conducting the attack cases.

Table 31: ADAssure Assertion Generation phase results.

Dataset		Assertion			Execution Time
Name	#Records	Total	$\#Next[\mathcal{N}]$	$\#Before[\mathcal{N}]$	
Longitude	412	5	3	2	1 ns
Latitude	356	7	7	0	1 ns
Delay	417	5	3	2	1 ns
GNSS	16	5	4	1	1 ns

Experimental Results

To evaluate the ADAssure methodology, we chose six attack types and their corresponding safety (benign) scenarios. These attack types included each of the aforementioned attacks with differing levels of noise (lateral and longitudinal position offset, delay message).

4.1.7.1 Automated Analysis

Utilising the ADAssure methodology on the three types of attacks yields three distinct set of assertions corresponding to each attack type. The results of the assertion generation phase are presented in Table 31.

The threshold for minimum support (`min_supp`) is set at 0.01 , while the minimum confidence (`min_conf`) threshold is 1 Notably, the method exhibits a swift execution time.

Within the 3 attacks of the cybersecurity corner case dataset, the assertions identify two patterns of anomalous AD behaviour. Firstly, extreme steering angles of 20° and -20° and sudden lane transition. Secondly, multiple lane-transitions combined with the extreme steering angle and sudden changes in vehicular velocity. This behaviour can be seen to be the effect of cyber activity on the smoothness of the initiation of the overtaking manoeuvre which results in turbulent movements and in some cases, a collision event. The assertions generated from the GNSS spoofing dataset identified the changes to the altitude and position co-variance. These were consistent with dramatic change in the values of the GPS coordinates and the resultant change in altitude.

4.1.7.2 Assertion Review and Debugging

The patterns identified in the association rules enables us to extrapolate that the Yaw angle and angular velocity are good reference point to show the effect of cyber-attacks. During the injection of the position offset attacks, the vehicle's orientation demonstrates dramatic action; in some circumstances, the vehicle can be seen to be essentially spinning. As displayed in Figure 45, the Lateral Position Offset Attack displays the Yaw (angle) of the vehicle making sharp changes, of 15 deg/sec from 15 meters mark of the AV journey. This vehicle dynamic behaviour is a characteristic also seen in both the longitudinal position offset (Figure 46) and delay message attack (Figure 47). The results for the velocity parameter demonstrate that it only indicates immediate collision of the vehicle, and it does not support early identification of anomalous vehicle behaviour. Assertion 1 contends that the AD system should not allow movements that challenge the physical limitations of the steering model.

Assertion 1: To determine the vulnerability of the yaw angle and momentum, we can derive the assertion: *AV.displacement_of_yaw_angle* $>$ *max_yaw_angle_threshold && time* $<$ *time_threshold*.

The roll-out transition, steer, and distance-to-collision parameters demonstrate identifiable change during a cyber-attack. The manipulation of the lateral and longitudinal position alters the vehicle position on the map and, therefore, has the effect of inducing greater transitions between roll-outs, which is the effective position of the vehicle on the road. The frequency of transition impacts the smoothness of the steering angle. From the distance-to-collision parameter, it is noted that the effect of the attack is most prominent during the overtaking maneuver and mostly during the cut-in process, when the vehicle cuts-in front of the passing vehicle (NPC). Assertion 2 contends that when the vehicle transitions across multiple roll-outs and displays 180° steering and closes to less than $0.5\,\mathrm{m}$ to the passing vehicle, this represents affected behaviour from the cyber attack.

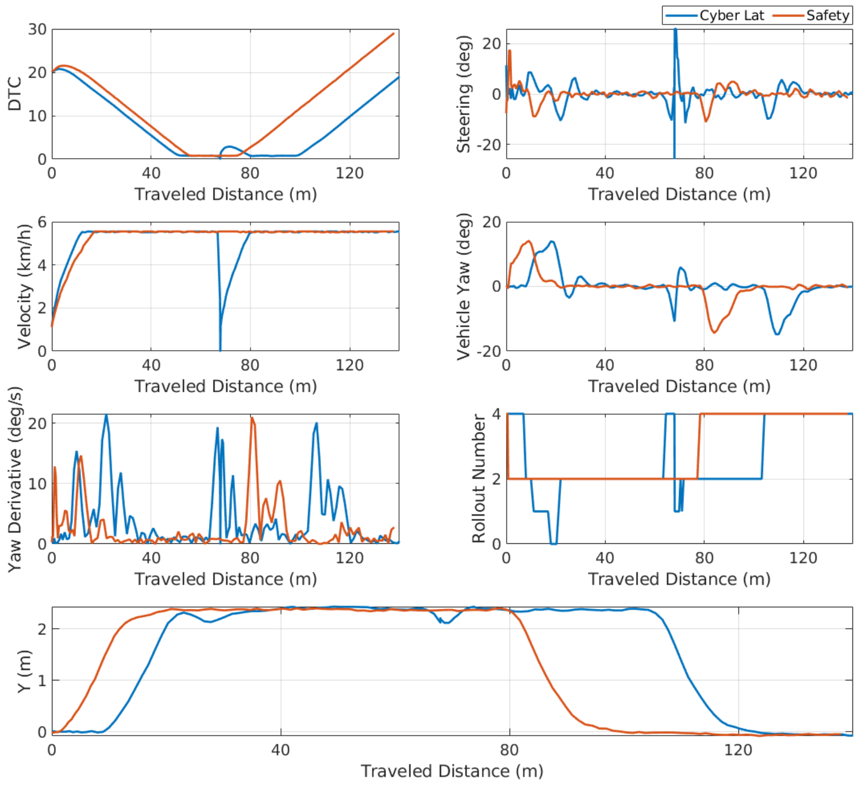


Figure 45: Lateral position offset attack vehicle parameters.

Assertion 2: To identify vehicle dynamic changes from cyber-attack: *AV.x* $-$ *NPC.x* $<$ *distance_threshold && AV.lane_transition* $>$ *max_transition_number && AV.steer_angle* $\notin$ *[min, max]_steer_angle*

Assertion 3 contends with activity seen in the longitudinal position offset (Figure 46) where the AV collides with the passing vehicle and then accelerates to the previous set-point.

Assertion 3: To identify collisions we can derive the assertion: $|AV.v_k - AV.v_{k+1}| > threshold$.

Assertion 3 could also be used to detect anomalies in GPS data. The GNSS spoofing attack demonstrates a significant deviation in the altitude and position co-variance parameters. Assuming that velocity data comes from two sources, a wheel sensor measurement and calculated by deriving the position from GPS data, the two results should be close to each other. In the case of a GNSS spoofing attack, the deviation in the position co-variance would generate a spike in the velocity (calculated by deriving the position in GPS data), and thus violating assertion 3.

For our specific AD system, the threshold for assertion 1 is 15° yaw angle displacement within $1\ \mathrm{s}$ duration. Assertion 2 threshold is identified as a distance between AV and passing vehicle as less than $0.5\ \mathrm{m}$, lane transition greater than 1 roll-out and steering angle that

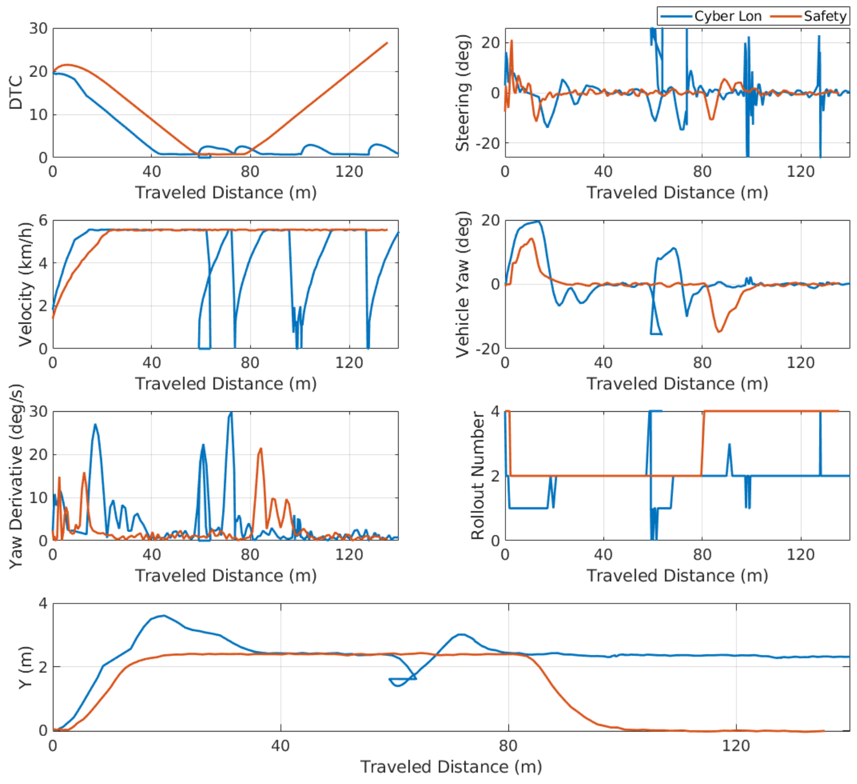


Figure 46: Longitudinal position offset attack vehicle parameters.

is outside the bounds of 20 and $-20°$. It is important to note that these values are valid for a low-speed AV ride-hailing service and for designers of different classes of vehicles, it is required to calculate values consistent with their specific application.

Solvable bugs come from several points in the controller; a simple one is wrong or imprecise saturation values of the control signal, which generates a high acceleration or a high steering angle in the vehicle. This is clearly visible in Figure 46 where a signal overshoot causes the vehicle to change lane multiple times. Another example, clearly visible in Figure 45,46& 47 is the lack of a fallback plan. There is a clear indication of a collision as the vehicle speed suddenly drops to $0\,\mathrm{m\,s^{-1}}$ and then quickly accelerates to the reference point, this is a violation of Assertion 3. A robust controller should have a fallback plan for such a case which indicates a bug in the functional design of the controller. In such a case, the vehicle should be aware of the fact that the global trajectory cannot be followed anymore and switch to emergency mode.

The main reason for searching for unexpected behaviours is to debug the controller, with reference to the experimental results, a violation of Assertion 1 can be associated to a bug in the */ndt_pose module* (see Figure 13), while a violation of Assertion 2 can be backpropagated to the module */op_trajectory_evaluator*. A violation of assertion three can be backpropagated to the modules of */op_trajectory_generator* and */op_behaviour_selector* (see Figure 13). To pinpoint the violation of assertion 3 to a specific function, we abstracted from the local_planner algorithm and its substituent lane_rule algorithm, the *getClosestWaypointNumber* method, which selects the next waypoint to follow in the global trajectory and returned an exception to be handled as a different driving behaviour (e.g., there

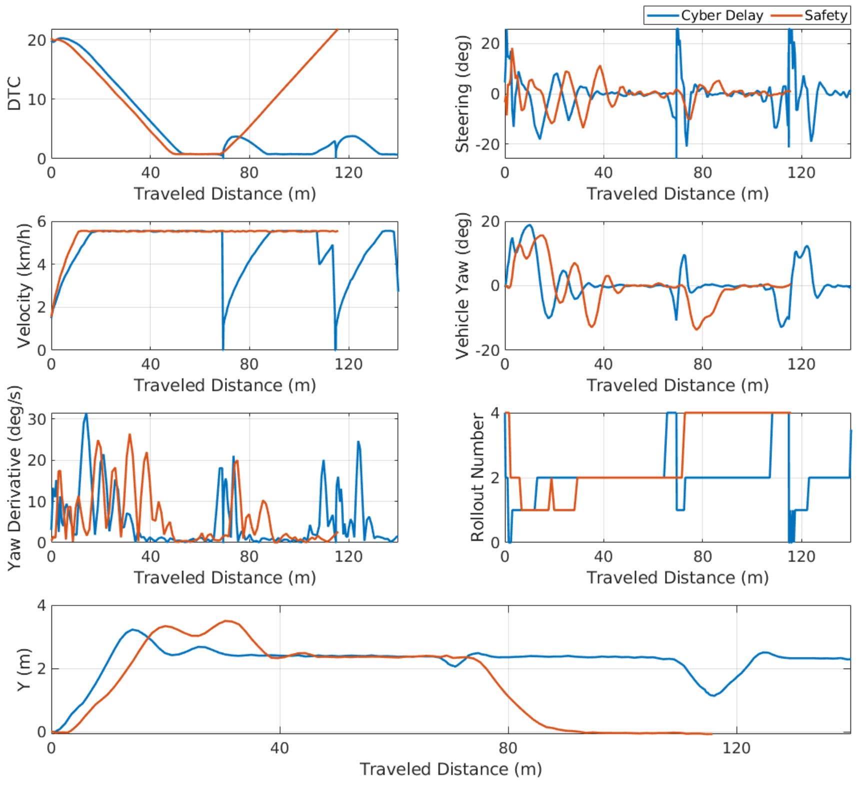


Figure 47: Delay message attack vehicle parameters.

was a crash, emergency mode activated).

In the case of GNSS attack, the NDT localisation algorithm doesn't detect the deviation in position co-variance, and this is due to the normal vector pointing in the same direction. Debugging focuses on optimisation of the NDT localisation using visual odometry for holding the local position at short-distances until the source of the disturbance has been resolved.

4.1.8 Relation to Existing Work

Recent publications on anomaly detection in vehicular AD control systems propose the usage of vehicle dynamics as a key detection indicator for cyber-attacks [140] [183] [262]. Studies such as Guo et al. [90] emphasise the effect cyber-attacks have on the trajectory of the AD system and the noise of individual sensors. Mitigation mechanisms focus on two diverse approaches 1) implementation of an observer of AD vehicle state estimation which can inform an emergency action (sensor switching etc.) [90] 2) implementation of trajectory smoothing algorithm to correct unplanned vehicle behaviour [183] [262]. However, these solutions for detection and mitigation are developed based on assumptions of driving environment and algorithm configuration and this limits the scope of their applicability.

4.2 REACT: Autonomous Intrusion Response for Intelligence Vehicles

In recent years, there has been remarkable progress in the development of smart vehicles. Today's vehicles resemble interconnected networks on wheels, with numerous embedded computers, called ECU, linked through various types of networks, hosting an extensive number of software components totaling over a hundred million lines of code. Moreover, these networks incorporate various intelligent sensors (such as Cameras, LiDAR, Radar, etc.) and different connectivity technologies that enhance the vehicle's ability to perceive and interact with the surrounding environment, thus bolstering autonomy and minimizing the reliance on human intervention. However, with the rise of connectivity and the transformation to SDV, the vulnerability to cyberattacks targeting these systems has also escalated [295].

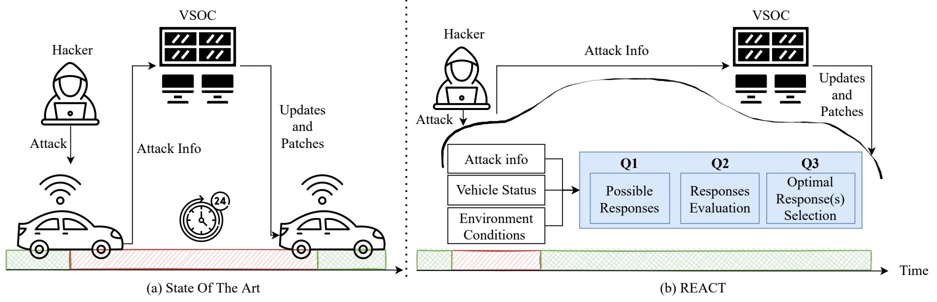


Figure 48: On the left side, the current vehicle system shares attack information with the VSOC but often has to wait for extended periods to receive necessary security patches and updates. This waiting period puts the vehicle in a malicious status (red, diagonal lines). On the right side, the vehicle can select and implement security solutions to avoid the long waiting time for security patches and updates and return to normal status (green, cross diagonal lines).

Recently, there has been a growing interest in addressing the security threats that may target smart vehicles. For instance, the ISO 21434 [123] standard has been introduced, with a significant portion dedicated to the development of threat analysis and risk assessment methodologies. Moreover, the field of intrusion detection and prevention in the automotive domain has witnessed extensive research, leading to various avenues for research [152]. However, despite these efforts, the number of attacks targeting smart vehicles continues to rise [295]. This is to be expected, as security is not absolute, and we must acknowledge that complete prevention of all security threats may not be attainable. Therefore, greater emphasis should be placed on defining *how the system should behave when confronted with such unavoidable attacks.*

The cybersecurity incident response is an integral aspect of security management, as outlined in ISO/SAE 21434 within the operational and maintenance clause [123]. Based on the standard, this process aims to provide remedial actions and updates, which may involve post-development changes to address security vulnerabilities. The process necessitates the vehicle to share cybersecurity information about the vulnerability that triggered the cybersecurity incident response. Being part of the ISO/SAE 21434, it is now imperative that manufacturers comply with new regulations by having a cybersecurity management system that oversees the cybersecurity activities and processes in the product life-cycle. To achieve this, Vehicle Security Operation Centers (VSOCs) will be utilized to support monitoring [23, 216, 257]. Such VSOCs will employ expert teams that continuously analyze data collected from all connected vehicles, enabling automakers to swiftly and efficiently address security incidents [216]. Although it's arguable that numerous tasks within a VSOC could be automated, the challenge of scalability persists, especially considering the extensive fleet of connected vehicles and the immense data volumes accumulated by each vehicle, reaching terabytes [314]. The transfer and processing of such data turn out to be significant issues, particularly in urban areas with hundreds of cars per vicinity, leading to bottlenecks. Additionally, the connectivity itself could be an attractive target for attackers. In this context, the integration of VSOCs into the smart vehicle ecosystem demands solutions for addressing connectivity challenges between vehicles and the VSOC, as well as managing privacy concerns tied to shared data [98].

Finally, and more importantly, there is a need to ensure a near-real-time response to security attacks. Taking into account the need for a human in the loop, as well as the latency introduced by high-volume shared data and communication between the vehicles and the VSOC, achieving a near-real-time response seems unrealistic. This perspective is supported by the European Union Agency for Cybersecurity (ENISA), which has cautioned that responding to high-criticality attacks could potentially take days or even weeks [75]. The scenario of extended waiting presents a dilemma, with two options, each having its own disadvantages. Allowing a vehicle to operate with a compromised component due to extended waiting for a security update is far from the ideal situation. Alternatively, suspending the compromised component until the security update is received might not be the best course of action either, particularly if the component plays a crucial role in operations.

Contributions: Therefore, there is a need for vehicles to be equipped with the capability to swiftly respond to cyberattacks. However, having such a capability requires the answering of three main questions (see Figure 48):

Q1: What are the possible responses that can be taken?

Q2: What factors need to be considered when evaluating these responses?

Q3: How to select one or more of these responses at the run-time based on the responses' evaluation?

This research aims to address these questions by investigating and categorizing potential responses according to the impact of various cyber attacks to which each response aims to react. Consequently, we present a dynamic risk assessment and cost evaluation for attacks and responses, utilizing given data such as attack information and vehicle status. This assessment supports the selection of suitable responses. Furthermore, the we explore different approaches for response selection, conducts comparisons, and identifies those best suited for automotive systems. We introduce an intrusion response system, referred to as REACT, and evaluate its utility using two attack scenarios. We evaluate the quality of the responses REACT generates and its overall efficiency. In summary, the main contributions of this paper are as follows:

4.2.1 Response Strategies

The purpose of this section is to address the first question (**Q1**) about possible response strategies. To do so, it is critical to have a deep understanding of the system as well as the potential attacks and threats it may face. Therefore, this section introduces the design of an automotive reference architecture, discusses the potential threats that may arise, and provides a comprehensive summary of the different response strategies that can be utilized to mitigate these attacks.

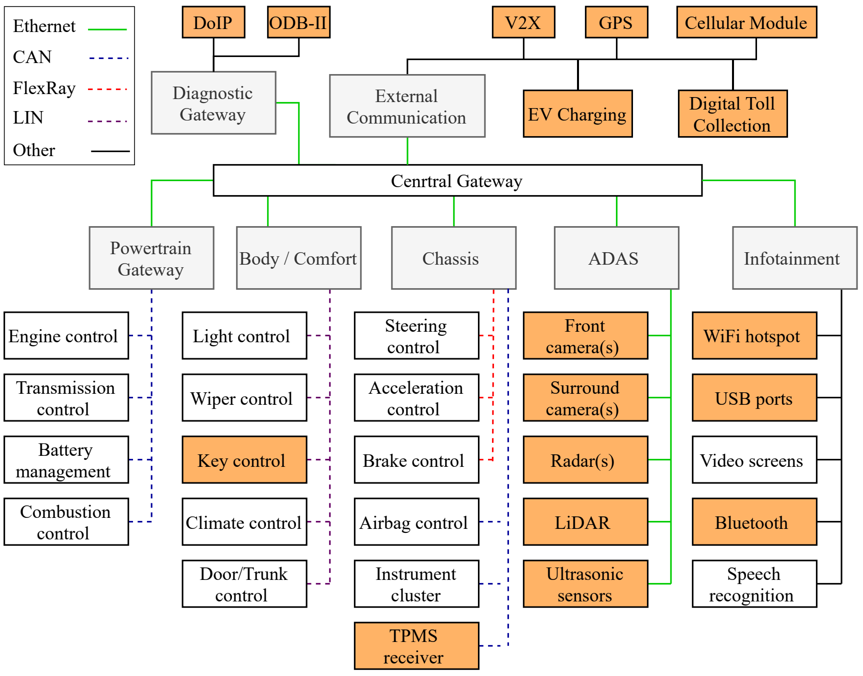


Figure 49: Reference vehicle architecture with possible attack surfaces (orange).

Automotive Reference Architecture

In order to understand how Intrusion Response System (IRS) can be integrated into modern vehicles and the potential responses they can provide, it is essential to first understand their system architecture. Figure 49 presents a generic, realistic and comprehensive reference architecture that can be found in modern vehicles. It is notable that a modern vehicle includes *highly interconnected* subsystems. The figure also shows how modern vehicles have many *embedded devices*, known as Electronic Control Units (ECUs), which are *distributed* allover the vehicle, communicating among themselves via different types of networks such as CAN, Flexray and Ethernet. These ECUs are grouped in different domains or zones based on the functionality such as infotainment, Advanced Driver Assistance System (ADAS), powertrains, etc. Besides ECUs, modern vehicles are equipped with many sensors (e.g., cameras, LiDAR, etc.), advanced communication technology for connecting with the external world, and diagnostic ports (e.g., OBD-II) that collectively form a significant attack surface for different types of attacks and threats [42]. The unrestricted or/and uncontrolled interaction among all those components puts the whole system in danger. Attackers could launch a *stepping-stone* attack [293], where they compromise a non-critical ECU with weaker security (e.g., the infotainment system), in order to gain control of a more crucial one (e.g., engine control) [53, 197]. All these characteristics of the vehicle architecture suggest that any proposed IRS should take into account the constrained resources and the highly interconnected and distributed nature of a vehicular system.

Threats and Attacks

Threat Analysis and Risk Assessment (TARA), an essential component of ISO 21434, is employed as a systematic way to identify and assess cybersecurity threats and risks in the

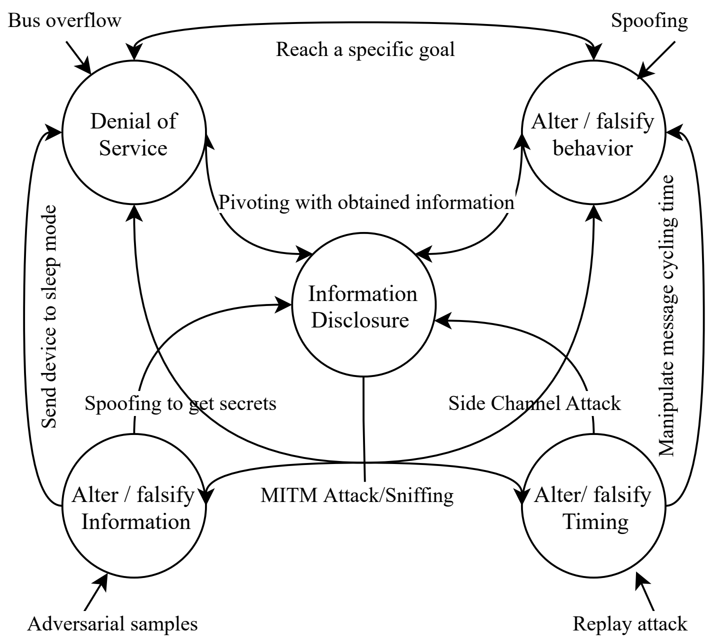


Figure 50: Classification of intrusion results and examples of attacks for each possible intrusion result.

automotive industry, facilitating the implementation of effective mitigation strategies. Since TARA does not dictate a specific method to identify threats, various methods have been proposed, such as STRIDE [142], SAVTA [97], attack trees [96, 109], and many others [179]. Following the methodology of TARA, these methods provide a comprehensive list of threats and attacks that may target the vehicular system and offer preventive measures. However, they do not address the reactive measures required for an automotive IRS.

Using the list of threats and attacks to create a response for each of them seems to be not ideal due to several challenges, including the large number of attacks and the requirements for precise information about each attack, which must be provided by the Intrusion Detection System (IDS). This challenge becomes evident when considering Zero-Day attacks, where information about such attacks may not be available to the IRS at the time of detection by the IDS. Even if an anomaly-based IDS shares some information about the attack pattern with the IRS, a response solely based on known attack patterns may not sufficiently react to these Zero-Day attacks. Therefore, the most effective approach is to enable the IRS to understand the situation it aims to respond to. This involves focusing on the impact or outcome of different attacks rather than solely on the attacks themselves.

To achieve that, we have developed a model, illustrated in 50, which represents the actual results of intrusions collected from various research works. The model encompasses five main attack outcomes, each of which can result from multiple types of attacks. Examples of these attacks are depicted in the outer nodes of 50. Also, to reflect the outcome of stepping-stone attacks, the model links the different outcomes to demonstrate that certain attacks may cause a series of results. The five attack outcomes are:

- *Falsify / Alter Information:* Different attacks have the potential to modify information on a bus or within an ECU. It is important to note that not every alteration of information automatically results in undesirable behavior. For instance, adversarial samples [184], such as incorrect classifications of objects detected by a camera, may not necessarily lead to incorrect behaviors.

- *Falsify / Alter Timing:* This outcome typically occurs as a result of attacks targeting the communication buses of the vehicle [175, 311] or the real-time tasks on the ECUs [95].

- *Information Disclosure:* This outcome is the result of attacks, such as spoofing, eavesdropping, and others, that aim to allow attackers to gain unauthorized access to sensitive information exchanged during communication or stored within the ECUs [54].

- *System Unavailability:* This outcome typically occurs as a result of Denial of Service (DoS) attacks that aim to cause a loss of availability for a specific component or subsystem in the vehicle [218]. Such attacks can lead to severe damage to the system, especially if they target high-critical components [9].

- *Falsify / Alter behavior:* This outcome is the result of tampering attacks that specifically target the components, data, or parameters of a system with the intention of altering the system's intended behavior and achieving unauthorized or malicious outcomes [197]. While this intrusion outcome may appear similar to falsify/alter information, the key distinction is that in falsify/alter information attacks, the goal is to tamper with the information itself without the explicit method of changing the system's behavior, even though it may indirectly lead to such changes.

Response Possibilities

After classifying the outcome of the attack, it becomes easier to determine which responses can be used to address that particular outcome and handle the attacks that cause it. In order to do so, we have examined typical responses discussed in both the automotive and non-automotive domains. It should be noted that while some research papers in the automotive domain have discussed the need for responses to certain attacks, there is currently no comprehensive research that lists and classifies all possible responses. Furthermore, it is important to consider that some of the responses we collected were originally designed for computer networks and may not be directly applicable to automotive bus systems due to the lack of specific security mechanisms [72]. For example, response actions such as IP address changes or port blocking [14] are highly specific to Ethernet and higher protocols such as IP, and therefore have limited suitability for certain aspects of communication in vehicles. To address this challenge, we have defined a list of generic responses that are specific enough to be applied in an automotive IRS, while also being adaptable to constrained and potentially insecure devices. Table 32 provides an overview of the different responses based on the identified attack outcomes. In addition, we have included a General category that encompasses responses applicable to all five categories. For more detailed information about each response, please refer to the respective sources cited in Table 32.

4.2.2 Dynamic Cost and Impact Evaluation

In this section, we will address **Q2** by outlining the key factors required to enable the selection of the most effective response by the IRS. These factors can be categorized into

Table 32: *Classification of generic responses to intrusion results.*

Intrusion Result	Response Index. Response
Falsify / Alter Timing	**1**. Use of redundant information [100], **2**. Correction of timing [72, 219], **3**. Force additional authentication [14], **4**. Restart the device/system [149], **5**. Change settings [117], **6**. Redirect traffic [117], **7**. Re-initialization [110]
Falsify / Alter Information	**1**. Use of redundant information (Reallocation) [100], **3**. Force additional authentication [14], **4**. Restart the device/system [149], **8**. Create a backup [49], **5**. Change settings [117], **7**. Re-initialization [110], **9**. Correct protocol specification faults [111], **10**. Split or merge functions [326]
Information Disclosure	**11**. Issue authentication challenges [219], **12**. Re-enforce access control [12], **3**. Force additional authentication [14], **13**. Introduce a honeypot [12], **4**. Restart the device/system [149], **14**. Modify firewall [117], **6**. Redirect traffic [117], **10**. Split or merge functions [326], **7**. Re-initialization [110], **15**. Network isolation [72]
System Unavailability	**1**. Use of redundant information (Reallocation) [100], **12**. Re-enforce access control [12], **13**. Introduce a honeypot [12], **4**. Restart the device/system (source or destination) [149], **14**. Modify firewall [117], **6**. Redirect traffic [117], **10**. Split or merge functions [326], **7**. Re-initialization [110], **16**. Limit resources of the attacker [49], **17**. Safe mode [99]
Falsify / Alter Behavior	**1**. Use of redundant information (Reallocation) [100], **18**. Correction of behavior [219], **9**. Correct protocol specification faults [111], **3**. Force additional authentication [14], **19**. Restart the miss-behaving system [149], **5**. Change settings [117], **10**. Split or merge functions [326], **7**. Re-initialization of the miss-behaving device [110], **17**. Safe mode [99], **8**. Create a backup [49]
General	**20**. Isolation [100], **21**. Limit communication of malicious system [100], **22**. Drop packets [149], **23**. Trace communication [100], **24**. Introduce additional logging [14], **25**. Block network traffic [12], **26**. Kill process [100], **27**. Reduce trust level of the source [100], **28**. Perform a security auditing [99], **29**. Request / Perform software update [219], **30**. Notify Security Operations Center (SOC) / administrator [12, 13], **31**. No action [13], **32**. Adapt parameters for IDS [108], **33**. Warn / inform other ECUs [19, 100]

two groups: *intrusion-related factors*, which pertain to the attack's impact and risk, and *response-related factors*, which concern the cost and benefit of the chosen response.

Intrusion-Related Factors

Intrusion Properties

For each detected intrusion, the following properties need to be determined:

- *Source of the intrusion:* This represents the component from which the attack was launched. Referring to the automotive reference architecture depicted in Figure 49, sources can include entities from the attack surface as well as external attackers targeting any of these components.

- *Destination of the intrusion:* The attacked entity can be described as the destination of the intrusion. This could be ECUs, sensors, or bus systems.
- *Intrusion result:* This refers to one of the outcomes that were previously defined in Subsection 4.2.1. Similar to the source and destination of an intrusion, this information is also provided by an IDS.
- *Intrusion impact*: This information serves to depict the impact of the intrusion on the system and is essential for evaluating the risks during the attack.

Dynamic Attack Impact Assessment

To assess the potential risks associated with an intrusion, it is necessary to understand the impact of the attack and the likelihood of its occurrence [123,168]. To calculate the impact of the intrusion, many methods were already adopted such as HEAVENS [126]. HEAVENS classifies the impact of a given threat based on four metrics [179,306]:

1. Safety impact, denoted as S with $S \in \{0, 10, 100, 1000\}$
2. Financial impact, denoted as F with $F \in \{0, 10, 100, 1000\}$
3. Operational impact, denoted as O with $O \in \{0, 1, 10, 100\}$
4. Privacy impact, denoted as P with $P \in \{0, 1, 10, 100\}$

In the original HEAVENS method, the overall impact I is calculated as a sum of the four single impacts as depicted in Equation 2 [306].

$$I = S + F + O + P \tag{2}$$

One issue with the impact calculation, as presented in Equation 2, is the overemphasis on safety and financial parameters. This skewed emphasis not only complicates the comparison and independent evaluation of the four metrics but also renders it unsuitable for an automotive IRS. In the automotive context, safety and operational considerations typically outweigh financial and privacy-related aspects for most automotive functions. Considering the aforementioned issue, we propose normalizing all possible values to $0, 1, 10, 100$, representing no, low, medium, or high impact for each of the four metrics in HEAVENS.

Another limitation of the current risk assessment methods, including HEAVENS, is their failure to account for dynamic environmental factors, such as run-time context, operational status, and the surrounding environment. This gap may arise because HEAVENS is primarily applied during the design phase, making it somewhat oblivious to run-time conditions. To address this challenge and enhance the method's applicability for use within automotive IRS, we introduce a new metric termed "Environment," denoted as E. This metric, E, encompasses dynamic factors that are crucial for assessing intrusion impact [100]. Potential inputs that can be used to derive the environmental parameter E include vehicle speed, road conditions, the proximity of nearby objects, and more. These parameters can exert significant influence, as a single intrusion may yield different impacts depending on physical and environmental considerations.

The final enhancement option for the HEAVENS method involves the capability to dynamically adjust the assessment of intrusion impact. Following a successful intrusion response, it may become evident that the stored parameters for S, F, O, P, and E require a different representation. HEAVENS currently confines impact values to $0, 1, 10, 100$, and a simple adjustment to a new value could result in significant over-representation. To address this issue, introducing weights for each of the five evaluation metrics (w_S, w_F, w_O, w_P, and w_E) offers a valuable mechanism for accommodating learning and adaptation processes. The optimization proposals discussed earlier to transform the calculation of intrusion impact using the HEAVENS method into a dynamic process lead to Equation 3.

$$I = w_S \cdot S + w_F \cdot F + w_O \cdot O + w_P \cdot P + w_E \cdot E \tag{3}$$

Utilizing dynamically adjusted static values for S, F, O, and P, each incorporating their respective weights, in addition to dynamically acquired values for E along with an adapted static weight. In cases involving specific automotive architectures, the equation can also be applied in a more granular fashion for particular assets. Initial values for all these parameters can be established by security experts, drawing upon their experiential knowledge.

The source and destination of the attack are employed to determine the attack's location, aiding in the calculation of the subsequent attack likelihood, especially when considering step-stone attacks, across various parts of the system. This assessment of attack likelihood, in conjunction with the evaluation of attack impact, contributes to the overall risk assessment.

Response-Related Factors

Response Properties

Similar to the intrusion, each response will have five properties that need to be identified:

- *Actual action:* They refer to the actual actions taken in the event of an intrusion. These actions can be selected from those presented in Table 32.
- *Precondition:* Some responses may require preconditions that must be met. These preconditions can be expressed as Boolean expressions and serve as prerequisites to trigger the response.
- *Place of application:* Refers to the location where the response will be implemented. A response can be applied either at the source entity of an intrusion, the destination, or at both locations.
- *Stop condition:* Refers to the condition for which the implemented response should cease. This condition can be related to a specific time [176], the successful reestablishment of security policies [100], or the necessity for persistent measures [293].
- *Cost and benefit of the response:* Refers to the costs and benefits incurred when implementing a response to an intrusion or security incident.

Dynamic response cost and benefit assessment

When considering the cost of responses, various methods were employed to determine their value in IT systems [261]. These methods primarily rely on one of three models: a static cost model that assigns a fixed cost value for each response, a static evaluated cost model that calculates cost using a static function with some adjustment possibilities, or dynamic evaluated cost models that offer fully dynamic evaluation based on real-time data. Each model varies in terms of simplicity, adaptability, and accuracy, catering to different system requirements and scenarios.

Statically evaluated cost models provide a valid trade-off between achievable implementation efforts, especially on constrained devices similar to the ones used in automotive systems, and plausible results. These models maintain a static approach to calculating response costs, even though the actual cost values may vary. Various metrics for calculating response costs are mentioned in current literature. The first metric evaluates the impact of the response on availability [261]. Availability's impact is represented as $A \in 0, 1, 10, 100$, with 0 meaning negligible and 100 meaning severe impact on availability, to ensure consistency with intrusion metrics. The second metric, describing the response cost, assesses its effect on the performance of the (sub)system [261], similar to the deployment cost of countermeasures [91]. This metric is denoted as $Perf \in 0, 1, 10, 100$, with 0 meaning negligible impact on performance and 100 meaning severe impact on performance, to maintain a uniform scale with the impact of the response on availability.

To achieve results similar to the adapted HEAVENS method described in 4.2.2, a comparable equation can be employed to calculate the cost (c) of a response. By adopting specific weights (w_A and w_{Perf}) for the impact on availability and performance along with their actual values (A and $Perf$), the response cost can be computed as shown in Equation 4. This approach results in a highly adaptable method for calculating the response cost. While the initial values for A and $Perf$ can be manually determined, they can also be adjusted over time. The specific weights offer a means to introduce a learning component within the mathematical framework.

$$c = w_A \cdot A + w_{Perf} \cdot Perf \tag{4}$$

Likewise, the adapted HEAVENS method introduced in 4.2.2 can be repurposed for evaluating the benefit of a response, with the exception of the environmental parameter E and its associated weight w_E. While HEAVENS assesses intrusion impact using four metrics, these same metrics can be employed to quantify the benefits in these four categories when assessing response value. By employing identical value possibilities with $S, F, O, P \in 0, 1, 10, 100$, a corresponding benefit value can be determined. The calculation of the benefit (b) for each response option, as shown in Equation 5, is derived from Equation 3.

$$b = w_S \cdot S + w_F \cdot F + w_O \cdot O + w_P \cdot P \tag{5}$$

Compared to existing research [91, 270], this repurposed HEAVENS method of Equation 5 provides a more holistic approach on evaluating the benefit of applied responses. For each response option classified in Table 32, the cost calculated using Equation 4 and the benefit determined using Equation 5 must be applied, and preconditions must be established. Initial values for S, F, O, P, A, and $Perf$, along with their respective weights, can be assigned by security experts and subsequently updated either manually or through learning algorithms within an IRS. Similar to the impact calculation of intrusions, these weights can be adjusted to improve the accuracy of the model.

4.2.3 Optimal Selection Algorithms

In this section, we will address the third question **Q3**, by exploring numerous potential methods for selecting response strategies(4.2.3), compare these approaches and provide a rationale for our chosen strategy(4.2.3), and describe how to adopt the selected strategies(4.2.3).

Possible Algorithms

To determine the best method for selecting appropriate responses, we explore various algorithms and solutions used in *non-automotive domains* and compare them to identify the most suitable one that can be implemented within the vehicle system. Several surveys, such as [24, 25, 211], provide valuable insights into response selection approaches in non-automotive domains, making them worth investigating for more comprehensive details.

Simple Additive Weighting (SAW)

SAW [80] is the simplest and most often used method. The basic concept of this method is to find a preference value (p) for each possible response, and then select the response with the highest preference value as the best option. To illustrate how this method works, let us assume that we have n possible responses ($\mathscr{R} = \{r_1, r_2, \ldots, r_n\}$) and m criteria ($\mathscr{CR} = \{cr_1, cr_2, \ldots, cr_m\}$) that will be used as a reference for evaluating the responses. Each criterion will be assigned a weight w_j where $\sum_{j=1}^{m} w_j = 1$. To calculate the preference values, a normalized decision matrix is first created, where each element of the matrix is normalized based on the nature of the criterion, whether it is a cost or benefit, as shown in Equation 6.

$$\alpha_{ij} = \begin{cases} \frac{v_{i,j}}{\max_i(v_{i,j})}, & \text{if criterion } cr_j \text{ is a benefit} \\ \frac{\min_i(v_{i,j})}{v_{i,j}}, & \text{if criterion } cr_j \text{ is a cost} \end{cases} \tag{6}$$

where $v_{i,j}$ is the performance value of the response r_i when it is evaluated in terms of criterion cr_j. The preference value (p_i) of response r_i is then obtained by calculating the weighted sum of the normalized performance values using Equation 7.

$$p_i = \sum_{j=1}^{m} w_j \cdot \alpha_{ij} \tag{7}$$

Finally, the response r_i with the highest preference value (p_i) is considered as the best selection response.

Linear Programming (LP)

LP is a mathematical technique that can be employed to select optimal responses [112]. LP can be used to find the best combination of responses that *maximizes* or *minimizes* a certain objective function. To illustrate the workings of this method, let us consider a scenario where we have n possible responses ($\mathscr{R} = r_1, r_2, \ldots, r_n$). The optimization of the objective function can be as in Equation 8.

$$\sum_{i=1}^{n} x_i s_i \rightarrow \max or \min \tag{8}$$

where x_i represents a criterion related to the response r_i and $\overrightarrow{s}$ be a vector of binary decision variables, where s_i is equal to 1, it indicates that the corresponding response $r_i \in \mathscr{R}$ will be executed. Conversely, if s_i is equal to 0, it signifies that the response $r_i \in \mathscr{R}$ will not be executed. The optimization problem typically includes *constraints* to ensure the selection process adheres to specific conditions or limitations.

Game-Theoretic Algorithm

Another mathematical method to determine optimal responses against cyber attacks is game-theoretic algorithms [299, 326, 342]. In the game-theoretic approach, the attacker and the IRS are modeled as two players. Each player has a set of actions available to them, such as different attack strategies $\mathscr{A} = \{a_1, a_2, \ldots, a_k\}$ for the attacker and response strategies $\mathscr{R} = \{r_1, r_2, \ldots, r_n\}$ for the IRS. The goal of the IRS is to select the optimal response to the attack at a given time. One way to achieve that is by minimizing the maximum damage of the attack: $\min_{r_i \in \mathscr{R}}(\max_{a_i \in \mathscr{A}}(U(r_i, a_i)))$ where $U(r_i, a_i)$ represents the utility function for the IRS when the attacker chooses attack a_i and the IRS responds with response r_i.

AI-based mechanisms

Many AI-based mechanisms were used to support the dynamic selection of the response such as Genetic Algorithms [78], Convolutional Neural Networks [318], Supervised machine learning [269], Q-Learning [120], and many more [243]. Using any of these AI models usually requires many steps including data collection and pre-processing, feature extracting, model training, and feedback loop to improve the quality of the selected responses.

Other Methods

There are alternative mathematical approaches to IRSs that are not derived from general mathematical problems. One example is REASSESS [217] that uses human-evaluated metrics and prior responses to select optimal responses. While it offers simplicity, this reliance on human evaluation can lead to inaccurate assumptions. Its mandatory learning behavior is unsuitable for automotive systems, and it lacks the option for flexible learning to enhance responses, requiring a well-established feedback loop. Another simpler approach is the cost-sensitive generic framework [271, 272], which includes steps like defining operational costs, ranking responses using a weighted sum method, and selecting the best response with an intrusion matrix. However, its reliance on static value assignments and sensitive parameters, typically defined by human experts, can make objective assessment challenging and results in potentially harmful responses.

Comparison

Table 33 summarizes all the advantages and the drawbacks of the five classes of response selection algorithms.

The primary advantage of SAW is its relative simplicity and utilization of lightweight mathematical operators, making it suitable for running on constrained devices with a polynomial run-time, without requiring complex external libraries [29]. However, the main drawback of SAW is the need for an adapted SAW method to achieve more accurate results. This often leads to increased complexity and longer run-time compared to the original SAW. Another drawback is the dependency on subjective parameters such as specific weights. This dependency can result in highly variable outcomes that may not accurately reflect the system state [160].

A major benefit of LP is its ability to formulate a single objective function and multiple constraints, providing an accurate representation of multi-objective optimization problems. However, compared to SAW, LP requires complex implementation, resulting in increased computational complexity for large systems [112]. The run-time of the algorithm depends on the solving method employed, such as the commonly used Simplex algorithm. While the Simplex algorithm has polynomial run-time for typical problems [253], it exhibits exponential worst-case run-time in theory [156].

The advantage of game-theoretic approaches lies in their consideration of the system state, resulting in a highly accurate representation of the system. Furthermore, game-theoretic approaches can be deployed in a distributed manner, as highlighted in [342]. A major drawback of this method is the use of highly complex models, which are necessary

Table 33: Comparison of the different response selection methods

Method	Benefits	Drawbacks
SAW	+ Simplicity and lightweight operators + Suitable for constrained devices + Polynomial run-time	- Adapted methods for accuracy increase complexity - Reliance on subjective parameters
LP	+ Flexible structures + Typically polynomial run-time + Existing libraries for solvers	- Higher complexity for modeling and calculation - Theoretically exponential run-time
Game-Theoretic Algorithms	+ System state consideration + Accurate system representation	- Very complex models - Computational complexity - Reliance on subjective parameters
AI-based Solutions	+ Handle large amount of data + Fast response selection	- Uncertainty of the selected responses - High resource requirements
Other Methods	+ Simple mathematical models + Typically fast + Combination with other methods possible + Learning is possible	- Complexity raises with large systems - Human influence has always subjective opinions

to determine optimal moves in game-theoretic algorithms. Solving such complex models often requires significant resources and leads to large communication overhead [342], making this approach unsuitable for constrained devices. Additionally, most models in practice make assumptions or simplifications due to the near-infinite number of possible system states [299, 326, 342], as complete modeling of all states is infeasible.

Using AI-based methods is still limited because of many issues such as the high memory and computation requirements of some of these methods [118] and the unrealistic responses that some models can produce (e.g., Genetic Algorithms). Additionally, uncertainty surrounding the outputs of these models limits their adoption. Finally, most of these methods rely on the availability of datasets for model training. However, autonomous vehicles often operate in dynamic and unpredictable environments. When the operating environment significantly deviates from what the AI has learned, it may encounter challenges in adapting effectively or making appropriate decisions.

Finally, while the cost-sensitive generic framework and REASSESS are simple and demonstrate promising in computer and network technologies, adapting them to a highly heterogeneous multi-bus architecture, like the vehicular reference architecture, presents significant challenges.

After careful consideration of the factors discussed above, we have chosen to explore the adapted SAW method, as well as LP with a focus on both benefit maximization and cost minimization for the design of an automotive IRS. The decision to focus on these two methods is based on their relative simplicity, computational efficiency, and their ability to accurately represent multi-objective optimization problems. The remaining algorithm families were assessed but are not pursued further due to reasons such as increased complexity, resource requirements, and limitations in modeling all possible system states.

Adopting of SAW and LP

Adopting of SAW

To adopt the SAW method for automotive IRSs, we first need to define the criteria $\mathscr{C}\mathscr{R}$ that will be used to evaluate each response. For this purpose, we can utilize the HEAVENS parameters, including the cost of a response c (see Equations 4) and the benefit of a response b (see Equation 5). However, using these two parameters still presents some issues that need to be addressed in order to effectively use and adapt SAW for valid results. The first problem arises when using these parameters during the creation of the elements of the normalized decision matrix, as depicted in Equation 6. This problem originates from the fact that our modified HEAVENS method allows values of $v_{i,j}$ to be in the set $0, 1, 10, 100$ for both criteria (i.e., c and b). If $\max_i(v_{i,j}) = 0$ applies, Equation 6 results in an illegal operation if the criterion is a benefit. Similarly, if the criterion is a cost and $v_{a,j} = 0$, Equation 6 also results in an illegal operation. This issue can be circumvented by using a small value greater than 0 instead of 0. The second problem does not stem from a mathematical perspective but rather from the application of this method in a fully automated IRS. Since the SAW method only considers criteria $\mathscr{C}\mathscr{R}$ from the applicable response set $\mathscr{R}$, it does not take into account the impact I of an intrusion. As a result of this limitation, it is possible that a response incurring high costs may be chosen even for a minor intrusion. Although this is a significant challenge for the application of SAW in IRSs, this drawback has not been addressed in existing research.

To tackle this problem, it is mandatory to set the preference value p (see Equation 7) into relation with the intrusion impact I. For each asset A of the vehicle reference architecture and each intrusion result $\mathscr{R}$, a normalized intrusion impact can be calculated. Such a normalized intrusion impact must be calculated for each metric S, F, O, P and E of the adapted HEAVENS method in Equation 3. This behavior is formulated in Equation 9.

$$[l]\alpha_{\{S,F,O,P,E\},A,\mathscr{R}} = \begin{cases} \frac{w_{\{S,F,O,P,E\},A,\mathscr{R}} \cdot v_{\{S,F,O,P,E\},A,\mathscr{R}}}{\sum_{|\mathscr{R}|}(w_{\{S,F,O,P,E\},A} \cdot v_{\{S,F,O,P,E\},A})}, & \text{if } \sum_{|\mathscr{R}|}(w_{\{S,F,O,P,E\},A} \cdot v_{\{S,F,O,P,E\},A}) \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

Similar to Equation 7, a weighted sum must be calculated. But, since the individual weights w are already included in Equation 9, a simple summation over all metrics S, F, O, P and E of the adapted HEAVENS method is sufficient. This sum will be set into relation with the preference value of the responses from Equation 7, such that the response r_i with the highest preference value p will be used, which is below the sum of all normalized HEAVENS values as depicted in Equation 10.

$$\text{best response} = \max\left\{ p_i \mid p_i < \rho \cdot \sum_{l \in \{S,F,O,P,E\}} \alpha_{l,A,\mathscr{R}} \right\} \tag{10}$$

The parameter ρ in Equation 10 is a parameter to adjust larger deviations in the order of magnitude between the sum of the normalized HEAVENS and the preference value p.

Adopting of Linear Programming

The first step to adopt the LP is defining the objective function. For the set of possible responses $\mathscr{R}$, it is possible to define two different objective functions:

- The first option of an objective function follows the principle of maximum benefit as depicted in Equation 11. The goal is to solve the binary decision vector $\overrightarrow{s}$ to maximize the benefit b. Although this can lead to very good solutions, it is possible that the best executable response is not found immediately since preconditions of identified responses are not satisfied.

$$\sum_{i=1}^{|\mathscr{R}|} s_i b_i \rightarrow \max \tag{11}$$

- The second option of an objective function follows the minimum cost principle and is comparable to existing IRSs [110,112]. Equation 12 therefore leads to more conservative responses since the cost c will be minimized and the benefit b of a response is not considered. A drawback is that the identified solution inside $\overrightarrow{s}$ might not heal the system completely and another try might be necessary.

$$\sum_{i=1}^{|\mathscr{R}|} s_i c_i \rightarrow \min \tag{12}$$

For both objective functions from Equation 11 and 12 the same constraints must be satisfied for a response to qualify for execution. Existing constraints of IRSs using LP [110, 112] are not suitable for an automotive IRS. Because of that, specific constraints must be elaborated:

1. The cost c of the response must be below the impact I of the detected intrusion [112]. Equation 13 depicts this first constraint.

$$\sum_{i=1}^{|\mathscr{R}|} s_i c_i < I \tag{13}$$

2. Only one response can and must be executed as depicted in Equation 14.

$$\sum_{i=1}^{|\mathscr{R}|} s_i = 1 \tag{14}$$

It is additionally necessary that $\overrightarrow{s}$ is a binary vector, leading to the variable definition $s_i \in \{0,1\}$.

4.2.4 Proposed Automotive IRS

In this section, we will discuss some design decisions regarding REACT, our proposed automotive IRS (refer to sec$_{i}rsdeployment)anddetailitscomponents(refertosec:irs_{c}omponents)$.

IRS Deployment

Our proposed automotive IRS can be deployed in three different locations:

- Central Gateway: The vehicle will have one IRS that receives information from various ECUs. This central IRS will have a comprehensive view and understanding of the entire system. However, it is considered a single point of failure.

- Domain Gateway: The vehicle will have one IRS per domain gateway. Each one will be mainly responsible for the ECUs belonging to that domain and will interact with other IRSs. Implementing this solution requires the existence of an Intrusion Response eXchange Protocol (IRXP) [100].

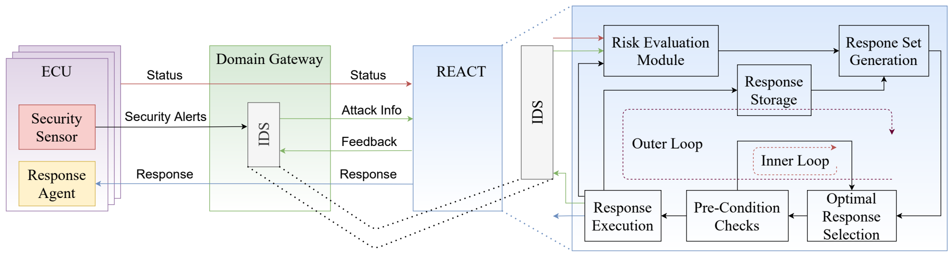


Figure 51: Internal architecture of REACT.

- ECU: The vehicle will have one IRS per ECU. This IRS will be primarily responsible for reacting to attacks related to its host ECU. Simultaneously, it can exchange responses related to other ECUs if needed. Choosing this option ensures the absence of a single point of failure. However, deploying such a solution requires that each ECU is capable of running the IRS, and it also necessitates the existence and the support of an IRXP [100].

The architecture depicted in Figure 51 illustrates the scenario where the IRS is deployed in the central gateway. Any potential change would be primarily associated with the source of certain information required for the functionality of the IRS, whether it originates from the same ECU (in the case of implementing the IRS per ECU) or from external sources such as other ECUs or domains at the gateway. Regardless of the chosen deployment location for the IRS, it necessitates the reception and sharing of information with other components within the vehicle, as outlined below:

- Attack Information: This information is provided by the IDS, and as described in 4.2.2, it includes the source of the attack, the destination, the intrusion result, and the impact of the attack. Recent IDSs, such as [66, 129], are capable of identifying the source and destination of an intrusion using various technologies, such as CAN databases (used by [129]) or ECU fingerprinting [50, 158]. The intrusion impact can be calculated as described in 4.2.2. Additionally, the intrusion result can be derived from the attack type, which existing IDSs, such as [105], can provide. In our research, we consider the IDS functionality as trusted, treating it as a blackbox that reliably detects intrusions without requiring additional false-positive handling [111, 292]. In our architecture, we place the IDS in the domain gateway. Consequently, a security sensor [13] is needed to monitor its portion of the environment for security-related observations. This data is then reported to the domain-specific gateway, which houses the domain IDS.

- Status Information: This includes information about the various states of the vehicle and its surroundings. This data is collected and aggregated from various vehicle sensors and shared with the IRS.

- Response Information: This information can encompass the precise responses needed for specific ECUs or those that need to be shared with the SOC. In our architecture, we assume the presence of response agents located in each ECU. These agents are responsible for receiving responses and deploying them within the respective ECU.

It is crucial to mention the necessity of ensuring the security of this data by implementing secure communication between the ECU, domain gateway, and the IRS.

4.2.5 IRS component

The IRS consists of the following sub-components (as shown in Figure 51):

- Risk Evaluation Module: This module will be responsible for assessing the impact of an intrusion. The component will receive information about the intrusion from the IDS as well as information about the vehicle status.
- Response Set Generation: This module compiles a list of possible responses, utilizing information obtained from both the IDS and the risk evaluation module. Please note that not every response is applicable to every type of intrusion result (refer to Table 32).
- Optimal Response Selection: This component integrates data from all previous modules to determine the optimal response that can be applied. Within this component, any of the algorithms presented in sec:posiblealgo can be integrated.
- Precondition Checking: Given the limitations imposed by the system architecture, where not all types of responses can be applied (for example, in cases where a sensor is unavailable due to a DoS attack, it may not always be possible to use a redundant source of information from another sensor if such a backup sensor does not exist), it is imperative to verify whether the selected optimal response is applicable or if an alternative response must be chosen. The Precondition Checking module receives the chosen response and assesses its feasibility. If a response is found to be inapplicable, a feedback loop is established with the previous Optimal Selection Module. This *inner loop* is repeated until the necessary preconditions for an individual response are met. The order of the Optimal Response Selection and the Precondition Checking is carefully evaluated and results in time benefits:

  1. "Check-First-Then-Select": The logical order of first eliminating all inapplicable responses and subsequently selecting the best response r from the remaining available options is illustrated by the timing behavior of Equation 15.

     $$t = \left(\sum_{i=1}^{|\mathscr{R}|} t_{check,r_i} \right) + t_{select,r} + t_{execute,r} \tag{15}$$

     The time to select the optimal response $t_{select,r}$ and the time to execute the response $t_{execute,r}$ are summed only once, since the selected response will satisfy the preconditions. In contrast, the time to check the preconditions $t_{check,r}$ is summed over the set of possible responses $\mathscr{R}$, since every response's precondition will be checked.
  2. "Select-First-Then-Check": While a response may be applied with the probability p, it might also be that the constraints are not satisfied with a probability $(1-p)$. This leads to a timing behavior of Equation 2.

     $$t = t_{select,r_1} + t_{check,r_1} + p \cdot t_{execute,r_1} + (1-p) \cdot \sum_{i=2}^{|\mathscr{R}|} \left(t_{select,r_i} + t_{check,r_i} \right)$$

     (16)

While the first selected response must always be checked, it is only executed with the probability p. If the preconditions are not satisfied, the *Inner Loop* will be repeated maximum $|\mathscr{R}|-1$ times.

It is evident that for a certain number of responses approaching infinity, Equations 15 and 2 yield the same runtime t when $p = 0.5$. For higher values of p, the runtime as per Equation 2 is even lower. This holds true even when $t_{select,r}$ decreases, as the number of possible responses decreases accordingly. Based on these equations, the architecture depicted in Figure 51 exhibits a "Select-First-Then-Check" behavior.

Response Execution: This component is responsible for transmitting the chosen response initially to the domain-specific gateways and subsequently to the respective ECUs for implementation through their local response engines. After a predefined duration, this component triggers the IDS to assess the effectiveness of the applied response in mitigating the intrusion. By incorporating this IDS-Feedback loop, the *Outer Loop* can be iterated multiple times, each iteration involving a system re-evaluation. This concept serves to counter persistent attacks or stepping-stone attacks effectively. Furthermore, the feedback loop can be utilized to update the parameters of the risk evaluation module for addressing future intrusions.

An essential consideration in the IRS architecture shown in Figure 51 is the implementation of termination criteria for the *inner* and *outer loop*. The absence of such criteria could lead to an endless loop, posing a risk to the stability of the entire IRS system. While some prior research has addressed termination criteria [100,261], these methods often involve complex evaluation techniques [38,119] or rely on artificial intelligence support [176]. However, the high computational requirements and intricate modeling approaches associated with these methods are impractical for automotive infrastructure. To address the challenge of preventing endless loops in both the *inner* and *outer* loops, we employ two distinct methods.

1. Preventing Inner Endless Loops: To avoid an endless evaluation of preconditions, we continuously reduce the possible response set by eliminating non-applicable responses. Additionally, we have introduced a special response, labeled as "No Action" (indexed as 31), which will consistently lead to the last possible response. This specific response carries the highest cost, similar to the impact of an intrusion, but provides no benefit. These attributes ensure that the *inner loop* never reaches a deadlock since "No Action" can always be applied.
2. Avoiding Outer Endless Loops: Once a response is applied, the system undergoes an analysis through the IDS-Feedback mechanism to identify if a new stepping-stone attack is detected or if the system is secure. In case a new stepping-stone attack is detected, the entire *outer loop* illustrated in Figure 51 reiterates. To prevent an endless loop scenario when the same response is repeatedly applied, we implement changes to the parameters of the applied response based on the success of the response. The parameter adaptation differs between a successful and a non-successful response. When the selected response is unsuccessful, it indicates that the benefit values assigned to all HEAVENS parameters may not be accurate. Consequently, an adjustment is needed, resulting in a reduction of the benefit values for all HEAVENS parameters in the previously applied response. This entails the assumption that the relative order of each parameter remains unchanged; for example, if the safety benefit held a higher value than the financial benefit prior to the adjustment, it will continue to do so afterward. This behavior is mathematically expressed in Equation 17.

$$\forall i \in \{S,F,O,P\}:$$

$$i_{\text{new}}(i_{\text{old}}) = \begin{cases} 10, & \text{if } i_{\text{old}} = 100 \\ 1, & \text{if } i_{\text{old}} = 10 \\ 0, & \text{if } i_{\text{old}} = 1 \text{ or } i_{\text{old}} = 0 \end{cases} \tag{17}$$

A similar parameter adaptation is required in case the response was applied successfully. However, the parameters cannot simply be increased, as this could lead to predictable responses. Predictable responses pose security risks, as attackers can exploit this behavior [29]. For that reason, two adaptations are made if the response is successful to avoid predictable behavior:

- Original values are restored if the response was previously not successful and its values were adapted according to Equation 17.
- In a second step, the corresponding weights $w_{i \in S,F,O,P}$ are randomly adjusted using a prefactor r, where $r_{min} \leq r \leq r_{max}$. This retains the original order of magnitude of w_i while introducing sufficient variation through the multiplication $r \cdot w_i$ to generate different results in the next iteration.

As previously mentioned, the parameters to calculate the intrusion impact (Equation 3), the response cost (Equation 4) and the response benefit (Equation 5) rely on input by security experts. However, this input may not always be optimal [168]. Consequently, this can lead to the selection of an undesired response. Fortunately, the *outer loop* provides a mechanism to compensate for potentially incorrect parameters. In cases where responses prove ineffective, the parameters are dynamically adapted using Equation 17.

Note that Equation 17 presented earlier does not account for the dynamic environmental parameter, denoted as E, and its corresponding weight, w_E. Further details and definitions are necessary to incorporate this parameter into the adaptation process. These details should encompass various aspects of the vehicle's status and its surrounding environment. For simplicity, we have focused on the vehicle's velocity as a parameter that can help represent the vehicle's status. To determine a realistic rating for the impact of vehicle speed, several factors must be taken into account. Studies of traffic accidents have revealed that the impact is influenced not only by the types of vehicles involved but also by their positions at the potential crash site [141]. Additionally, the age of the passengers in the vehicles can affect the impact of injuries in a traffic accident [231]. Based on this research, the approach presented in Equation 18 is applied to the parameter E in the adapted HEAVENS method's prototype implementation [141, 231].

$$E(v) = \begin{cases} 100, & \text{if } v \geq 75\ km/h \\ 10, & \text{if } 50\ km/h \leq v < 75\ km/h \\ 1, & \text{if } 30\ km/h \leq v < 50\ km/h \\ 0, & \text{if } 0\ km/h \leq v < 30\ km/h \end{cases} \tag{18}$$

Response Storage: Within this component, a repository is maintained containing a range of potential responses alongside their associated metrics. These metrics can be updated through the feedback mechanism or expanded with the inclusion of new responses and parameters via an external connectivity interface. When implementing this on specific hardware, it is crucial to implement security measures to prevent unauthorized tampering with the memory area.

Our proposed IRS architecture, featuring both an *inner loop* and an *outer loop*, coupled with the incorporation of automotive-specific considerations into the external architecture, introduces a novel paradigm in the realm of fully automated IRSs. Note that there is already some related work for each part of the IRS (such as the selection method), which was covered in the previous sections. However, there is no system that attempts to include all the aspects against which we can compare our work.

4.2.6 Evaluation

Table 34: IDS-related information and vehicle state parameters for both evaluation scenarios.

Property	**Scenario 1**	**Scenario 2**
Name	Adversarial sample	Information disclosure at the infotainment system
Infected Asset	Front Camera	Infotainment Gateway
Affected Asset	Acceleration control	Infotainment Gateway
Intrusion Result	Falsify / Alter behavior	Information Disclosure
Dynamic Parameter	Velocity: 70 km/h	Velocity: 0 km/h

Implementation, Testbed, and Use Cases

The proposed IRS was implemented using the Python programming language. To implement Linear Programming and the associated Simplex algorithm, we utilized the `PuLP library` [199], a well-established choice, along with the GNU Linear Programming Kit as the solver. It is important to note that the adapted SAW method remains independent of this decision, as it relies solely on standard Python mathematical operators.

The testbed designed for evaluating the IRS incorporates an embedded system setup to realistically emulate the automotive infrastructure. To ensure this fidelity, our implementation was executed on a Raspberry Pi 4 Model B Rev 1.2, a choice justified by the device's ARM-based quad-core processor running at $1.5\,\mathrm{GHz}$. This processing power closely aligns with the high-performance chips commonly found in the automotive industry.

The goal of the evaluation is to assess two key aspects of the proposed IRS. Firstly, we aim to evaluate its proficiency in optimal response selection, and secondly, we intend to measure various computational metrics, including memory consumption and the time required to obtain optimal responses while using the three different selection algorithms: LP with maximum benefit, LP with minimum cost, and adapted SAW.

For our evaluation, we employed two representative intrusion scenarios inspired by real-world intrusions:

1. Adversarial Sample: This scenario involves slight modifications to the input data of a machine learning algorithm, resulting in significantly different outputs from the original [184]. Given the prevalent use of machine learning algorithms in cameras for automated vehicles, they are vulnerable to exploitation via adversarial samples [184]. In our evaluation, we exploited a front camera in a rural setting, leading to an altered behavior in the acceleration control.

2. Information Disclosure at the Infotainment System: This scenario draws inspiration from an actual attack on a vehicle, where an information disclosure in the infotainment system served as the initial step in a stepping-stone attack [197].

The specific IDS parameters and vehicle states employed as input for the scenarios are meticulously detailed in Table 34. Please remember that in our prototype of the IRS, we consider only the velocity of the attacked vehicle as an illustrative example of a vehicle's status.

4.2.7 Results

In this section, we will present the results of testing our IRS using two prominent scenarios. We will evaluate response quality, response selection time, memory consumption, and the adaptation of response parameters for each of the three selection algorithms: LP with maximum benefit, LP with minimum cost, and the adapted SAW.

Response Quality

The objective of the response quality evaluation is to assess how different optimal selection algorithms prioritize responses and determine the overall impact and benefit of the applied responses. To achieve that, the precondition of each response is set to 'rejected' for every proposed response. This ensures that the IRS will continue to suggest responses from the list of possible responses. Each applied response can have both positive and negative effects on the system, so the cost and benefit values of the selected responses are presented. In this evaluation, default parameters are utilized for each new test, ensuring uniformity in the algorithm evaluation across various metrics.

Figure 52 depicts the cost and benefit of all proposed responses in the order they are applied by the respective algorithm for both scenarios. The figure shows that our proposed IRS suggests a different number and order of responses for various scenarios and for different selection algorithms within the same scenario. Please note that the figure shows that some responses were selected twice. For example, the response of restarting the misbehaving system (indexed with number 19, see Table 32), was selected twice. However, it is important to clarify that the response was selected for different systems. In other words, the first restart is related to the camera, while the second is for the acceleration control. In addition, as expected and shown in Figure 52(a) and Figure 52(b), the LP method with maximum benefit starts at very high benefits. Similarly, the LP with minimum response costs starts at a very low cost and more expensive responses are not selected until later stages, as shown in Figure52(c) and Figure 52(d). Notably, the LP with maximum benefit operates independently of the cost. However, it always ensures that the cost of the response is less than the impact of the intrusion (see Equation 13).

The reason for the arbitrary behavior is that Linear Programming only follows one optimization function and just satisfies the constraints, but does not sort by constraints. Similarly, LP with minimum cost delivers arbitrary values with respect to the benefit because it only considers cost metrics in its optimization. While the LP with the minimum cost provides more conservative solutions, the LP with maximum benefit suggests more offensive solutions. In a real-world scenario, LP with minimum cost might require multiple responses since its benefits are arbitrarily sorted, while LP with maximum benefit might require more iterations of the "inner loop" since the preconditions for more offensive responses might not be fulfilled.

The adapted SAW method exhibits a similar arbitrary behavior as shown in Figure 52(e) and Figure 52(f). However, it is noticeable that adapted SAW may select responses with a cost higher than the impact of the intrusion (see Figure 52(f)). Given that the adapted SAW method does not consider constraints, it is an unattractive solution to use any SAW method in an automatic IRS.

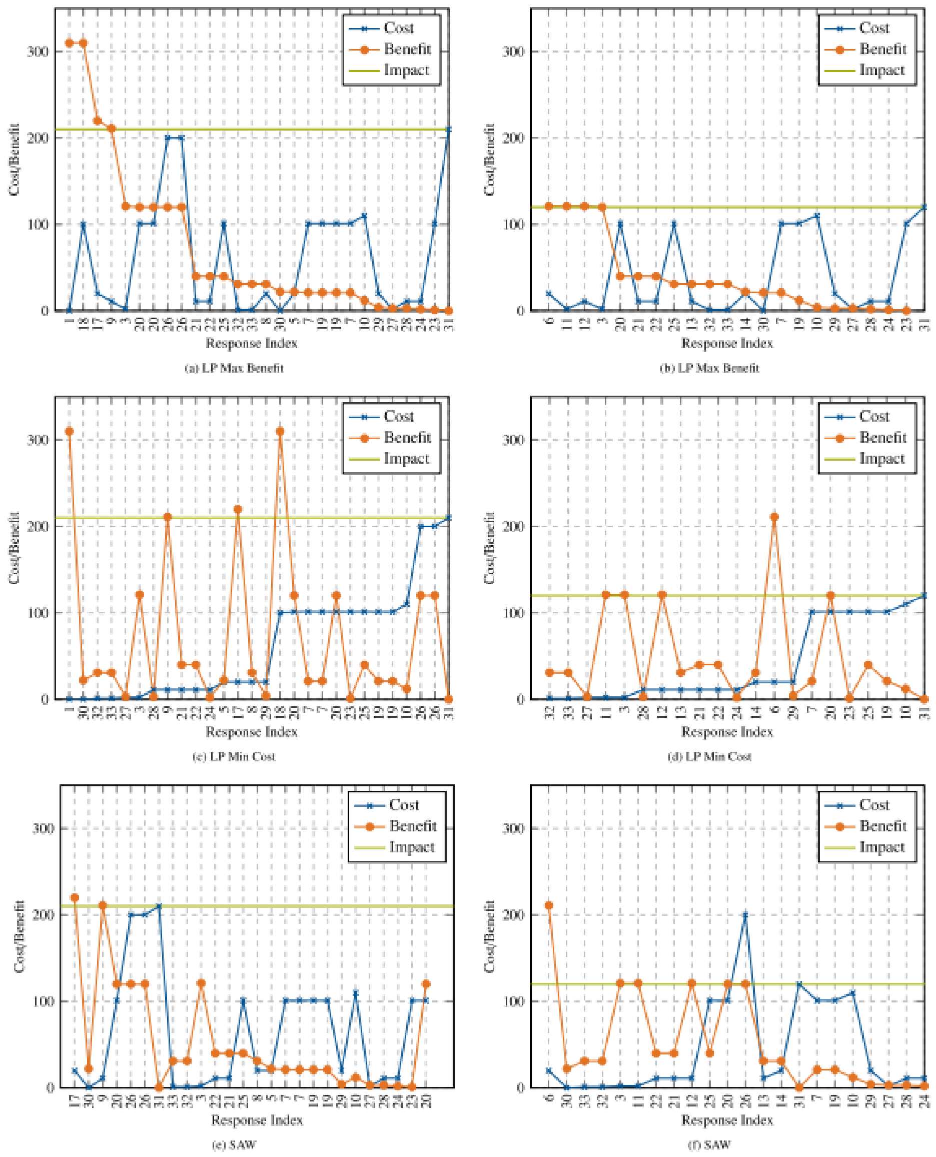


Figure 52: Evaluation of the response benefit and cost for Scenario 1 (left) and Scenario 2 (right) using LP with maximum benefit (top), LP with minimum cost (middle), and adapted SAW (bottom)

Time of Response Selection

To evaluate the time required for selecting a response from a given response list using the selection algorithms, we utilized the previously described method where the *inner loop* of the IRS repeats multiple times. It is important to note that the generation of the response set occurs only once for an individual intrusion. The time required for list generation is independent of the selection algorithm, measuring at $4.32\,\mathrm{ms}$ for scenario 1 and $3.82\,\mathrm{ms}$ for scenario 2. The difference in the measured time between the scenarios is due to the variation in number of possible responses.

Figure 53 illustrates the time consumed by the three selection algorithms during the

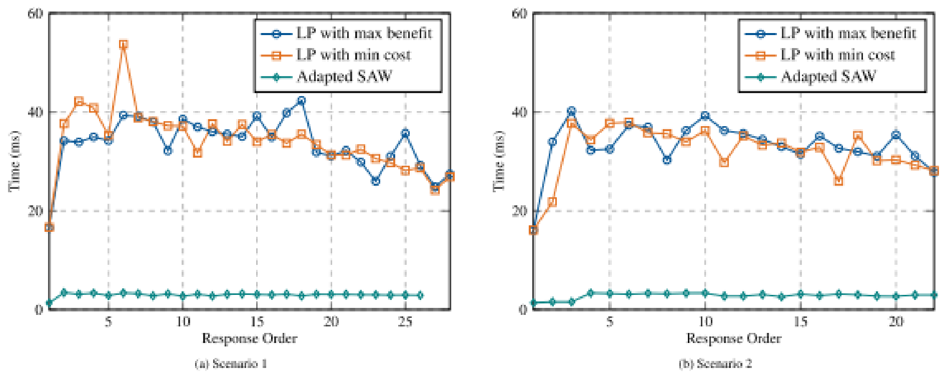


Figure 53: Evaluation of consumed time for response selection using the three selection algorithms for both scenarios

Table 35: Memory consumption of the IRS in kB using static evaluation.

	LP with Max Benefit	LP with Min Cost	Adapted SAW
Scenario 1	19308	19206	11296
Scenario 2	19228	19344	11220

process of selecting different responses. Please note that the X-axis represents the order of the response, not the index of the response. The figure indicates that the adapted SAW method consumes less time compared to the LP methods. Specifically, the LP method with maximum benefit typically consumes more time due to the need for multiple iterations, as its offensive responses may not meet necessary preconditions. Slightly less time is needed for the LP method with minimum cost, although its conservative responses are selected after fewer precondition checks. Overall, all algorithms demonstrate good performance on a resource-constrained embedded system.

Memory Consumption

To measure memory consumption, we utilized Python's internal `resource` module [227]. Since some of the optimal selection algorithms rely on third-party libraries, the assessment of memory consumption includes the memory allocated for these functionalities as well. The results are presented in Table 35. The results show that both LP with maximum benefit and LP with minimum cost methods consume nearly the same amount of memory, while the adapted SAW method exhibits considerably lower memory consumption. This difference can be attributed to the external libraries `PuLP` and the `GNU Linear Programming Kit`, which require more memory due to their complex data structures and solving methods. Nevertheless, all three selection algorithms exhibit low memory consumption, making them suitable for use in resource-constrained embedded hardware systems.

Dynamic Evaluation

The dynamic evaluation concentrates on two key aspects: response and threat impact parameters adaptation (refer to 4.2.2) and the inclusion of velocity considerations (as shown

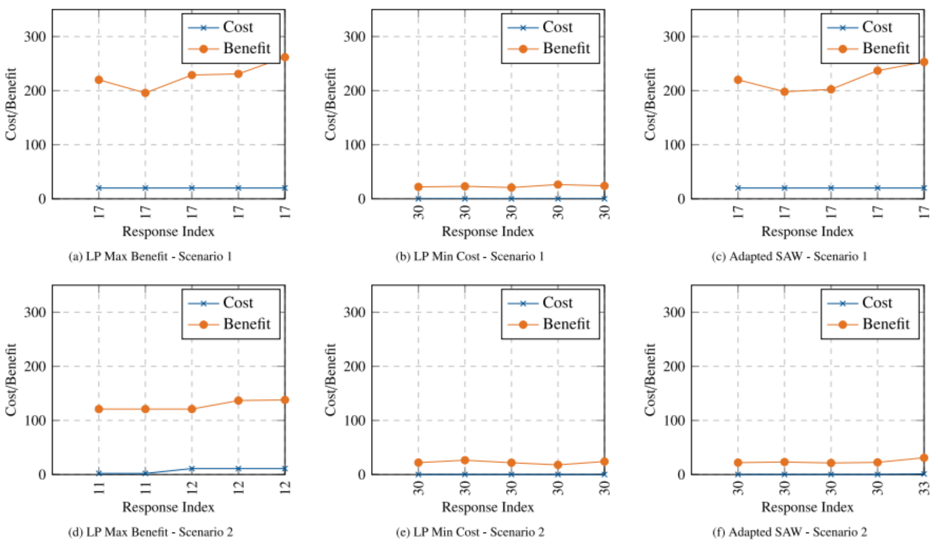


Figure 54: Evaluation of parameter adaptation in Scenario 1 (top) and Scenario 2 (bottom) for the responses selected over five iterations using the three selection algorithms, assuming the responses were consistently considered successful.

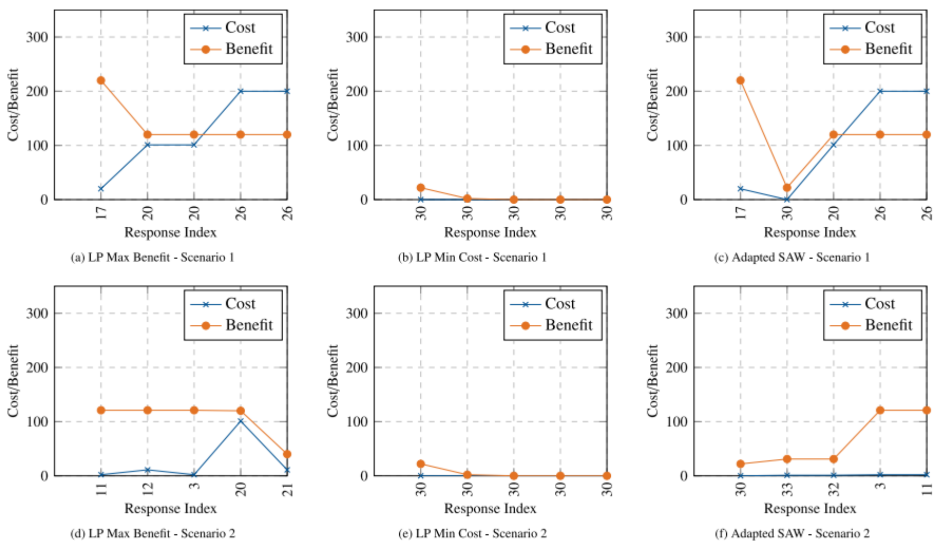


Figure 55: Evaluation of parameter adaptation in Scenario 1 (top) and Scenario 2 (bottom) for the responses selected over five iterations using the three selection algorithms, assuming the responses were consistently considered unsuccessful.

in Equation 18). When it comes to parameters adaptation, response quality is assessed based on their cost and benefit. In terms of velocity, we evaluate response variation. These assessments are conducted for both scenarios 1 and 2. By testing all three implemented optimal selection algorithms, we can compare their dynamic behavior.

4.2.7.1 Parameters adaption To assess the impact of changing parameters, we conducted two repetitions of each scenario, each comprising five iterations of the *outer loop*. In one set of iterations for each scenario, we consistently deemed the responses as successful, while in the other set of five iterations, the responses were uniformly considered unsuccessful. The benefits and costs of the five optimally selected responses for both scenarios, as determined by the three selection algorithms, under the assumption that the responses were always successful, are presented in Figure 54. Correspondingly, the results under the assumption that the responses were consistently unsuccessful are displayed in Figure 55.

In consistently successful attacks, we observed that parameter weights change within the range of ±20% (we have selected $r_{min} = 0.8$ and $r_{max} = 1.2$). The purpose of these changes was to reduce response predictability. In both scenarios, changes in response benefit were evident. However, in the first scenario, all three algorithms retained the same response as shown in Figures 54(a), 54(b), and 54(c). This was changed in the second scenario, where responses were altered for the LP with maximum benefit and adaptive SAW algorithms as shown in Figures 54(d), and 54(f). The reason for the absence of changes in the selected responses in the first scenario when using LP with maximum benefits or adapted SAW algorithms can be attributed to the specific response chosen: transitioning to a safe mode (indexed with 17). This response had very high benefit values, as determined through the initial evaluation process, making minor variations of ±20% inconsequential to the overall result. Consequently, minor variations of ±20% did not affect the overall result, as the next possible response had significantly lower benefit values. To avoid such a constant behavior, a more substantial modification of the response parameters or the use of an asymmetric window for the prefactor, with a higher probability of negative values, can be implemented. Notably, the LP method with minimum cost (Figure 54(b) and 54(e) did not consider response benefits in its optimization function, rendering modifications to response benefit irrelevant. This method-related limitation persisted across both simulated scenarios.

In the case of consistently unsuccessful attacks, we observe more substantial variations in the selected responses compared to the previous case (see Figure 55). This behavior is expected, as the parameter adaptation in a non-successful case involves higher orders of magnitude, as shown in Equation 17, compared to the successful case. Similar to the previous analysis, the LP method with minimum cost optimization consistently generates the same response due to the exclusion of response benefit in the optimization process, as shown in Figures 55(b) and 55(e). Conversely, LP with maximum benefit optimization aligns with expectations. Although the initial response is similar to the successful case, subsequent responses exhibit lower benefits (Figures 55(a) and 55(d)) and higher costs as a side effect. Notably, response index 26 (killing the process) appeared twice in Figures 55(a) and 55(c), each referring to different components (i.e., camera and acceleration control). The adapted SAW method consistently produces varying results with less distinct trends in benefit and cost when compared to LP with maximum benefit (Figures 55(c) and 55(f)). This observed behavior holds true for both scenarios 1 and 2, underscoring the expected functionality of parameter adaptation for non-successful cases.

In conclusion, this assessment of dynamic parameter adaptation confirms that LP with maximum benefit and the adapted SAW methods perform effectively with adjusted parameters, rendering the results valid for both test cases. On the other hand, the LP method with minimum cost optimization falls short in its capacity to respond to parameter shifts in response benefit values. Consequently, this method appears less appealing for identifying optimal responses in autonomous IRS.

Table 36: *Impact of the velocity for the evaluated scenarios, using Equation 3.*

	Impact (unitless)		
	0 km/h	**50 km/h**	**100 km/h**
Scenario 1	200	210	300
Scenario 2	120	130	220

4.2.7.2 Inclusion of Velocity Considerations The second key aspect of dynamic evaluation involves assessing the influence of vehicle velocity on the selected responses. In our current prototype system, the environmental parameter E is treated similarly to other HEAVENS parameters in Equation 3, as their respective weights w are either one or zero. As we alter the velocity, the environmental parameter for an intrusion takes on different values, as indicated in Equation 18. Therefore, intrusion's impact is more significant at higher velocities. For this test, both scenario one and two are assessed at three velocities: 0, 50, and 100 km/h, using all three implemented algorithms, with each evaluation beginning with the default data-set.

While the intrusion impact calculation in Table 36 functions as expected, each algorithm consistently selects the same response within each scenario, regardless of the velocity. This behavior can be attributed to the high impact values in the two evaluated scenarios. In cases of less severe intrusions or during the early stages of a stepping-stone attack, where the HEAVENS parameters result in lower values, the velocity's impact becomes relatively more substantial, thus leading to varying results. Nonetheless, it's important to emphasize that the proposed IRS architecture is adaptable since the individual weights w for HEAVENS parameters can be customized as per Equation 3. This customization minimizes the over-representation of static HEAVENS parameters, enabling the velocity to exert a more pronounced influence on the selected response.

Final Remarks

The evaluation of the developed IRS reveals the advantages and drawbacks of each selection method. The adapted SAW method is limited by its inability to consider constraints. Consequently, it is not feasible to employ this method in a fully automated IRS. On the other hand, LP with minimum cost consistently favors constant responses and is, therefore, unsuitable for optimal response identification. Despite its successful application in existing research [110, 112], the results demonstrate suboptimal behavior for the automotive use case. Nevertheless, it is well-suited for proposing follow-up responses once the primary intrusion has been mitigated. These follow-up responses can enhance security by alerting a SOC and providing information to the car manufacturer, ultimately leading to updated software. In contrast, the LP method with maximum benefit, excels in all metrics evaluated for an automotive IRS. Since it offers responses with high benefits from the outset, it is well-suited to respond to the primary intrusion.

4.2.8 Conclusion and Outlook

Modern vehicles' intricate architecture and advanced connectivity present unique intrusion challenges. While automotive security research has traditionally emphasized IDSs as a secondary defense layer, the development of vehicle IRS is in its early stages, drawing inspiration from related industries. To delve into the development of an automotive IRS, we sought answers to three key questions: defining potential responses, outlining response evaluation criteria, and optimizing response selection. Initially, we categorized automotive intrusions and stepping-stone attacks into five distinct categories to create a more versatile intrusion model. Similarly, we classified responses, creating a formal description for both intrusions and responses. Additionally, we investigated necessary adjustments to existing risk assessment models to support response evaluation. Furthermore, we conducted a comprehensive comparison of various optimal selection algorithms, highlighting the adaptability of the SAW method and Linear Programming (LP) with various optimizations for IRS integration. Although other algorithm families may gain relevance in the future, they currently face limitations in the automotive context. In addition to these findings, we proposed an IRS architecture that accommodates the distributed nature of vehicles and addresses automotive-specific constraints. Evaluation in real-world scenarios has led to the development of a novel vehicular IRS, demonstrating its potential for integration into modern distributed vehicle architectures and enhancing overall security.

While the focus of the paper is on the analysis and design of the IRS, the implementation of the external architecture and the response execution modules on the local engines on each ECU is still a challenge towards an IRS as a system. To test such an overall IRS system, real-world data sets, including both normal operation and attack scenarios, are needed. Extensive evaluation in Software-in-the-Loop or Hardware-in-the-Loop testbeds can extend the existing evaluations of algorithms and the overall system. With respect to the secure communication of intrusions and responses, further research and standardization are needed to be performed in order to ensure that the developed IRS does not only reply in an adequate manner but also distributes its responses. In this direction, leveraging existing efforts such as [124, 193] by extending them towards establishing a standardized method for securely exchanging the proposed responses within the vehicle and with other vehicles would provide a solid foundation, as these existing standards and guidelines already offer valuable insights. Also, it is important to note that the functionality of our proposed system depends on the availability of information about the attack, such as its source, destination, and type, which needs to be provided by the IDS. This information can be obtained by integrating existing research approaches, as demonstrated in [66, 129]. Finally, the modular architecture of REACT allows an easy extension towards more complex vehicle architectures and new intrusions or responses. Additionally it allows the integration of new selection algorithms in the future to adapt to possible changed needs.

4.3 Summary

Within this section, we developed methods for fingerprinting, debugging and intrusion response in AD systems. The ADAssure methodology leverages vehicle dynamics data for automated and manual analysis of indicators of activity of cyber attack. ADAssure demonstrated that indicators of attack could be found through vehicle dynamics data and that assertions on the system could be generated. These assertions can aid in the improvement of the design of the control system software. The analysis of IRS techniques for automotive consequently demonstrates the difficulty in finding an optimal solution for a real-time, safety-critical system with timing and resource constraints. Both analyses highlight the complexity of development of cyber defensive mechanisms given the vast data ecosystem and system-of-system environment. Within the next section we will conversely explore the development of tools for cybersecurity testing of AD software.

5 Tools for Autonomous Driving Software Cybersecurity Testing

5.1 Cybersecurity Test Range for Autonomous Vehicles

AV shuttles for public transportation are being piloted in European cities [121]. Cybersecurity of AV shuttles is of predominant importance for the safety of passengers and pedestrians in the traffic environment. Digitisation of vehicles and the transitioning to intelligent control by algorithms have exposed vulnerabilities to traditional cyber attacks such as ransomware, distributed denial of service, and new attack surfaces such as adversarial machine learning and sensor manipulation [43, 220, 223, 280]. Recent examples [7, 11] of software failures of semi-autonomous vehicles resulting in fatalities of passengers have shown the lethal potentiality of cyber attacks. There are many challenges to securing AV shuttles against cyber attacks.

Cyber ranges are popular tools to experiment with edge and corner cybersecurity test cases and training for skills development and situational awareness of cybersecurity incident response. However, there is a lack of evaluation of cyber range technologies for AV cybersecurity and knowledge as to how cyber-physical systems can translate to support real-world, operational AV shuttles.

To address the challenges of AV cybersecurity, cybersecurity testing platforms for cyber-physical systems and methods for testing and training are required. In this research we evaluate the Massachusetts Institute of Technology (MIT) Duckietown, low-cost, small-factor, cyber-physical AV test bed to support cybersecurity testing of a real-world AV Shuttle, operating in Tallinn, Estonia. The purpose is to understand how a cyber-physical test bed can be used for cybersecurity testing of AV shuttles and how this can transform cyber ranges and training for AV cybersecurity. The main activities of this research are the following:

1. We investigate the utility of a cyber-physical test bed for AV shuttles to support a real-world, operational AV shuttle.

2. We demonstrate, through a series of practical cybersecurity test scenarios, that a low-cost, cyber-physical test bed can be used to test the general cybersecurity of an AV shuttle and improve issues with the architecture and training for situational awareness of operators.

3. We outline recommendations how a cyber-physical test bed can be used to validate cybersecurity edge and corner cases.

5.1.1 Relation to Existing Work

Cyber-physical test beds for AVs have featured in numerous studies. However, the related work is focused on the design of the test bed and there are few works that include considerations for cybersecurity testing and training.

Three studies are prominent in the related literature for their focus on designing low-cost cyber-physical test beds for automotive.

Axelsson *et al.* created a vehicle test bed for security evaluation of cyber physical system. The test bed was based on a small-factor mobile vehicle which was customised to support AUTOSAR, a software framework for automotive. The vehicle test bed, developed in 2014, demonstrated that a small-factor device could provide a solution to emulate the protocols and features of a full-factor real-life vehicle. The test bed was not autonomous and relied on remote control by human operator [21].

Tian developed a low-cost cyber-physical AV for research of neural networks. The research involved creating a code base for a line following car in a low-noise, controlled, test environment. The study developed the environment which could have applications for test bed and cyber range capability. However, as this was not the primary focus of the study, the translation of the cyber-physical AV for testing was not explored [70].

Bhadani *et al.* created a Cognitive and Autonomous Test (CAT) Vehicle test bed to evaluate AVs. The research problem highlighted in the study was the cost, time and risks of real-world testing and the problems translating test cases from simulators to real-world environments. The study designs and builds a hybrid virtual-physical test bed that incorporates the body physics of a real world vehicle with virtualised sensors and software platforms. ROS is used as the middleware platform. The evaluation of the platform was conducted through an educational program where students used extracted data from the CAT vehicle to improve object detection and tracking. The study was focused on the design of the vehicle and not cybersecurity, testing or training [26].

Zelle *et al.* and Santos & Schoop extended the design of a test bed for AVs to include a framework for cybersecurity testing of AVs. Both of these studies focused on test cases generated from either formal methods or system analysis [67, 329]. Zelle *et al.* built a security test platform for AVs using small-factor cyber-physical systems. The methods used in designing the platform comprised eliciting an attack model of cybersecurity attacks against autonomous vehicles. Based on this attack model the test bed was designed. The test bed is innovative, it includes most of the diverse range of sensors used for perception as well as in-vehicular networks and infotainment systems. The contribution is closest to this work. The main differentiation is that this study provided a practical assessment of the test bed and analysed testing and training methods that a cyber-physical AV test bed could support [329].

Santos & Schoop developed a framework for cybersecurity testing of AVs and evaluated its efficiency through investigation of the survivability of autonomous vehicles after a cyber attack to the vehicles sensors. Their framework consisted of developing test cases from formal methods and a tool to auto-generate test cases. Their practical evaluation involved the security testing of two sensors; camera and LiDAR. An open-source autonomous driving simulator, CARLA, was used as the experimental testing environment. The authors tool for automatic test case generation only supports CARLA. Their study acknowledges the limitations of this approach, the attack to the sensors was delivered by manual scripts and assumed the attackers could manipulate the sensors perfectly each time. The findings are limited to the CARLA environment and the simulation environment testing could not replicate a real-world physical attack or the operational driving domain of the vehicle [67].

5.1.2 AV Shuttle Cybersecurity Program

To select a low-cost, cyber-physical test bed to evaluate for AV shuttle cybersecurity, we begin by providing an overview of the Tallinn, Estonia, iseAuto, a real-world, operational AV shuttle.

AV shuttles are a type of AV used for public transportation in predominantly urban environments. AV shuttles can accommodate up to 15 passengers and are limited is speed to approximately 25 km/h. Table 37 lists a few of the different models of operational AV shuttles. There are thousands of AV shuttles currently operating around the world [121]. Figure 56 depicts a public transport AV shuttle.

Table 37: AV Shuttles for Public Transportation [121]

AV Shuttle	Location	Environment
Navya Arma	Parc Olympique Lyonnais, France	Public Road
EasyMile EZ10	Airport Velizy-Villacoublay, Paris, France	Government Property
iseAuto	Tallinn, Estonia	Private Road
Baidu Apollo	Software Park Xiamen, China	Public Road
Local Motors Olli	Goodyear, Colmar-Berg, Luxembourg	Private Road

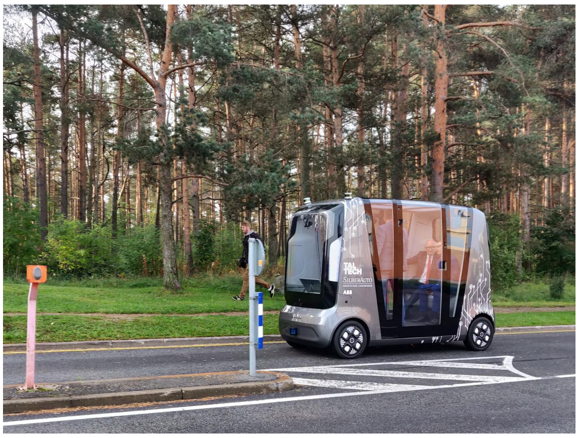


Figure 56: iseAuto Public transport AV Shuttle [16]

AV Shuttles use either open-source or proprietary software designed for AVs. ROS is one of the key open-source systems. ROS is an open-source middleware that provides support for hardware abstraction, low-level device control, message-passing between processes, and package management. ROS is popularly used as it integrates with Autoware, a large open-source research and development community that provides a software platform for autonomous driving. The Autoware platform provides modules for self-driving, these include localisation, detection, prediction, planning and control [**?**]. These modules are essential for the vehicle to understand where it is located in the driving environment, map the route it must drive and detect the objects in the driving environment such as pedestrians. Furthermore, the control module is crucial for the vehicle to coordinate the conditions under which the control of the vehicle will be maintained and important decisions will be made, such as when control of the vehicle will be passed back to the human operator.

The AV shuttle architecture integrates this software ecosystem with advanced hardware technology and sensors: LiDAR, ultrasonic radar, camera and GNSS.

AVs use teleoperation. Teleoperation is the remote monitoring and controlling of the AV by a human operator. In the real-world vehicle used in this study, the teleoperation is a software module of the ROS, enabling communication between the on-board computer and a remote teleoperation centre located in a building along the private road environment.

AV Shuttles are densely interconnected. The internal self-driving vehicle network consists of layers of communicating devices from the embedded components of the vehicle, including the ECUs using the CAN Bus protocol, to the IP connected sensors. The vehicle communicates with smart road-sign-units (RSUs) and internet-connected devices, which is termed V2X, and with other vehicles, known as V2V.

The AV shuttles autonomous driving cognition and sensonics are tested in simulators and cyber-physical test beds. Popular simulators include; Apollo Baidu, LGSVL, CARLA and ROS Gazebo [241]. Simulators consist of a 3D generated driving environment, normally from the maps generated by LiDAR sensor. The simulated AV can take as input the same configurations used in the ROS software of the real-world vehicle and similar sensor software profiles. Cyber-physical test beds can be either small-factor replicas or hardware-in-the-loop test benches. Cyber-physical test beds allow the same features and functionalities of the simulated environment with the additional benefit of providing testing of physical interfaces and the dynamic of real-world physical conditions.

In 2015, researches demonstrated that the in-vehicle network, Controller Area Network (CAN) Bus, of a Jeep Cherokee could be exploited through malware and remote code injection, to stop the brakes of the vehicle [55]. This event precipitated the increase in focus on testing methods and test platforms for CAN Bus and connected vehicle technologies; communication interfaces and infotainment systems. This increase in research activity has lead to an increasing amount of vulnerabilities found in connected vehicles (Table 38).

Table 38: Examples of Cyber Attacks on Connected Vehicles

Vehicle	**Cyber Threat**
Tesla Model S	Spoof Passive keyless entry to take advantage of weak cryptography, lack of mutual authentication for challenge-response and lack of firmware protection [313]. Malicious firmware with linux kernel exploitation for the ConnMan open-source internet connection manager allows WiFi of the Tesla to be exploited to allow remote connections [307].
Jeep Cherokee 2014	Malware on Infotainment system to allow remote root privileges and pivot into CAN Bus network [55].
KIA	Reverse-engineered Android OS Infotainment system. Found vulnerabilities to allow remote root privileges [82].

There has been growing research in cybersecurity vulnerabilities of autonomous driving. These mainly focus on adversarial machine learning that aims to exploit weaknesses in the autonomous driving cognition, fuzzing of ROS and other middleware software, and network interfaces used for V2V and V2X communication (Table 39). Most of this research is conducted in simulators or on isolated systems and components and very few of the testing methods relate to real-world operational vehicles [43,195,220,223,230,280]. An exhaustive list of vulnerabilities of connected vehicles and AVs can be found here [148]

Table 39: Examples of Cyber Attacks on AVs

Attack Surface	Cyber Threat
Autonomous Cognition	Tamper with RSU Stop Sign to manipulate autonomous cognition [267].
	Tamper with lane markings to manipulate lane-keeping system(LKAS) [249].
	Spoofed images in driving environment to manipulate object-detection [210].
Sensors	Jam LiDAR point cloud sensor with laser [35].
	Tamper with sensor data to manipulate navigation path [61].
System	Spoofing of ROS Master and interception of ROS messages [130]
	Malware in firmware update [308]
	Fuzzing of AV middleware [103].
Network	Intercept and spoof RSU messages [223]

Despite commonly used regression testing methods and standards for cyber assurance testing of AVs, the vulnerabilities of AV systems continue grow.

Public transportation AV shuttles undergo limited testing for cybersecurity, this is due to many reasons. Firstly, cybersecurity testing on real-world proving grounds with operational vehicles is expensive and time-consuming, requiring extensive labour effort in the setup, execution and safety monitoring of the tests [297]. Secondly, there is a reluctance to test cybersecurity test cases that could damage the vehicle. This is mainly due to the cost and time involved in rebuilding and re-configuring vehicular systems and components. Thirdly, the AV shuttle architecture is a distributed systems architecture and due to lack of testing there is a gap in understanding how cyber attacks cause cascading affects and how, for instance, malware could propagate throughout the system. Fourthly, there is a lack of investigation of novel testing methods and techniques for cybersecurity. These include software simulators and cyber-physical test beds commonly used for testing autonomous driving cognition. Lastly, there is lack of training of teleoperation, remote control vehicle operators for situational awareness for cybersecurity. As AV shuttles rely on teleoperation operators to override the autonomous cognition in emergency situations and make manual driving decisions, their awareness as to how cyber attacks can impact situational awareness is critical for safe driving operation.

Flexible testing environments that allow agile testing of edge and corner cyberseecurity test scenarios would help assist in identifying vulnerabilities of the AV system architecture. Whilst simulators and small-factor cyber-physical test beds are used for testing and improving autonomous driving algorithms there has been limited practical exploration of these test beds for cybersecurity testing.

Test beds such as the MIT DuckieTown, provide a low-cost, small-factor environment accessible to autonomous self-driving vehicle developers and quality assurance testers [278]. These environments, which utilise the same software and network interfaces as Autonomous Vehicle (AV) Shuttles have the potential to be used for cybersecurity testing and research.

5.1.3 Cybersecurity Test Beds for AV Shuttles

Key factors which influence the design and usage of test beds to support the operational AV shuttle include *cost*, *complexity* and *fidelity* of the test bed to the operational system.

Cost

To support agile testing and cybersecurity test cases that impart physical damage on the AV, the cost of the test bed needs to be as limited as possible. The low-cost requirement has two intended beneficiaries. Firstly, a low-cost agile test bed can be given to students and researchers in innovative testing formats such as crowd sourcing. This enables a wider scope of testing for minimal cost. Secondly, AV shuttle programs for public transportation do not have exhaustive resources for testing in comparison to the major original equipment manufacturers. Therefore, low cost test beds are required to test edge and corner cases and prioritise test cases for testing on the real-world vehicle.

Complexity

AV shuttles are a complex distributed system architecture, it is essential that the test bed support observation of distributed system interaction whilst limiting the complexity to allow rebuilding of damaged systems. For example, allowing a clean rebuilding of a software or hardware system infected by malware. This agility will allow repeatable testing of cybersecurity test cases and enable dynamic testing such as crowdsourced vulnerability analysis and training such as capture-the-flag style learning activities.

Fidelity to Operational Vehicle

To evaluate cybersecurity and situational awareness there needs to be a level of abstraction of the operational vehicle architecture. An evaluation of the real-world AV shuttle considered the AI & Drive systems, sensonics and the network connectivity with the teleoperation as key features of the autonomous driving architecture to emulate in a test bed.

Test Bed Analysis

A comparison of test beds used for autonomous driving and cybersecurity research found the cyber-physical test bed to be an optimal platform for evaluation (Table 40). Advantages of the cyber-physical system are the low cost and agile, modular architecture which can allow sensors and systems to be added or removed. Due to the lack of evaluation of cyber-physical test beds to support cybersecurity testing their fidelity to real-world, operational system is yet to be determined, and will be explored in this study. Whilst real-world proving grounds offer the highest fidelity, they come with a considerable cost due to resources required to engineer tests with real vehicles and manage the safety risks of such tests.

Table 40: Comparison of Test Bed Architectures to support Cybersecurity

Testing Considerations	**Simulation**	**Cyber-Physical**	**Real-World Proving Ground**
Cost	Low	Low	High
Complexity	Low	Medium	High
Fidelity	Medium	Not evaluated	High

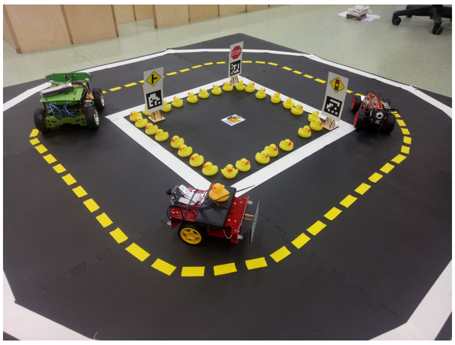
Table 41 presents an evaluation of the test bed architectures to support testing for the cyber attacks listed in Table 38 and Table 39.

Table 41: Comparison of Test Bed Architectures to support AV Shuttle cybersecurity Test Cases

Cyber Threat Test Cases	**Simulation**	**Cyber-Physical**	**Real-World Proving Ground**
Hardware and Compute	Yes	Yes	Yes
Connected Vehicle	Yes	Yes	Yes
Sensor and Perception	Yes	Yes	Yes
Physical Access	No	Yes	Yes
Damage Incurring	No	Yes	Yes
Environmental Perturbations	No	Yes	Yes

AV Shuttle Cyber Range for Cybersecurity

The MIT CSAIL Duckietown is a small-factor test bed used for evaluating autonomous driving software modules, algorithms and education. Duckietown consists of a driving environment (Figure 57a) and an AV, called, DuckieBot (Figure 57b).The cost of the components to build the MIT Duckietown test bed is approximately €400.


(a) DuckieTown - Cyber-Physical Test Bed

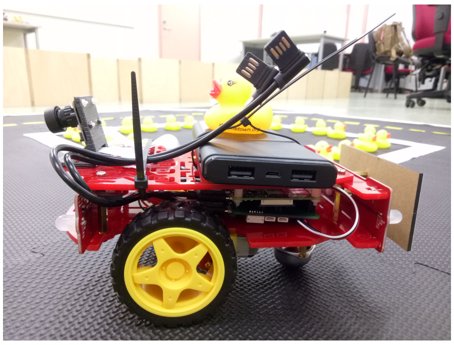

(b) Duckiebot - AV Cyber-Physical Device

Figure 57: MIT DuckieTown Self-Driving TestBed

The DuckieBot uses a 5 mega pixel Raspberry Pi camera for sensing. The hardware for the AI and Drive Algorithm is built on Raspberry Pi Model 3B hardware. The software platform is built upon Docker utilising ROS Kinetic. A 32GB SD card is used for local on-board storage and a 100Gb USB drive can be inserted in the Raspberry Pi to allow more storage for logging. A 5 volt, 10400 mAh, battery is used to power the DuckieBot. Actuation is performed by the motor driver which connects to servo motors. The DuckieBot is calibrated to operate in the DuckieTown driving environment. This consists of a floor layer with road markings, conventional to the standard markings of real-world traffic.

Table 42 represents detailed analysis of the DuckieBot with the iseAuto AV shuttle, operating in Tallinn, Estonia.

Table 42: Feature Comparison of Test Bed and iseAuto AV Shuttle

DuckieBot	**iseAuto AV Shuttle**
ROS Kinetic Kame	ROS Kinetic Kame
Linux Network Interfaces and 4G Cellular Network	Linux Network Interfaces and 4G/5G Cellular Connectivity (*V2X is yet to be added as a feature)
Camera Sensing	Camera, LiDAR, Ultrasonic Radar, GNSS
Actuation, motor driver controls servo motors	Actuation, Drive Controller controls CAR ECU
On-board Control PC (ARM processor)	On-board Control PC (ARM processor) different hardware specifications
Teleoperation - Mission Control System	Teleoperation - Mission Control System

The DuckieBot is an optimal test bed for experimentation as it uses the containerised architecture of Docker. This allows software packages for sensors, hardware and AD to be centralised in a configurable system. This enables packages to be added or removed depending on the test case and for new sensors and hardware to be added easily. The other major advantage is that the DuckieBot is an actively supported open-source project and new AD algorithms are published regularly. This helps to ensure that test cases are tested against the newest available AD algorithms.

5.1.4 Cybersecurity Test Scenarios for AV Shuttles

Test Scenario Generation Process

Selected use-cases are used to evaluate the usefulness of the cyber-physical range. To generate the cyber test scenarios we asked experts in AV cybersecurity from vehicle manufacturers and system designers to detail edge and corner cybersecurity test cases that they would want a AV cybersecurity test bed to support. The experts represented organisations that develop autonomous robots for logistics, autonomous driving assistance systems and AV shuttle operators. Table 43 lists illustrates our chosen demonstration use-cases.

Table 43: Security Test Case Scenarios

Test Case	**Description**
Scenario 1 (S01)	An external threat actor spoofs the road markings to manipulate the driving logic to veer the vehicle off the road.
Scenario 2 (S02)	An external threat actor tampers with the road markings to manipulate the drive logic to veer the vehicle off the road.
Scenario 3 (S03)	An external threat actor tampers with the camera sensor using a laser pointer to blind or shield it's perception to manipulate the drive logic to veer the vehicle off the road.
Scenario 4 (S04)	An external threat actor spoofs the RSU to manipulate the drive logic to veer the vehicle off the road.
Scenario 5 (S05)	An innocent maintenance engineer executes a malicious cryptocurrency or ransomware malware hiding as a firmware update for a vehicle system.
Scenario 6 (S06)	An external threat actor eavesdrops on the ROS vehicular messaging system for information gathering
Scenario 7 (S07)	An external threat actor attacker conducts a denial-of-service of the teleoperation communication link with the AV.
Scenario 8 (S08)	An external threat actor uses a smoke gun to perturb the camera sensor vision and alter the driving course of the vehicle

Scenarios

The aim of the test scenarios is to understand the utility of cybersecurity testing in an AV cyber-physical testbed environment to the real-world AV shuttle. The verification of the test results does not focus on a binary, yes/no conclusion, rather, a deeper analysis of whether the behaviour of the AV system observed during the cybersecurity testing can be used to identify vulnerabilities of the real-world AV shuttle architecture to cyber attacks. All of the scenarios can be viewed at the YouTube channel that was created to demonstrate the AV cyber range: `https://tinyurl.com/2xxvvkzd`

S01 - Projected Road Markings

Problem: The projector attack consists of an attacker crafting a spoof image to be projected onto the traffic environment. The aim for the attacker is to fool the autonomous drive cognition to accept the spoofed image as genuine and alter the driving behaviour. An example would be a project of a lane marking on the road to alter the course of the AV. The projection attack experiments as detailed in Nassi *et al.* [210], used trial-and-error as a method of testing. The attack was trialled on real-world vehicles in a private campus environment. The testing environment was tightly controlled for safety reasons and the setup of the test took considerable time and effort. In DuckieTown, this attack can be tested and repeated using as many diverse methods as possible. The small, cyber-physical testing environment allowed for agility and repeatability and enabled replication of a cyber threat identified in a paper to test the validity of the results to our Av shuttle.

Whilst a spoofing attack using projection is a novel and interesting method to manipulate an autonomous vehicle it is unlikely or has low probability of success. The projection must contend with natural light, which means the attack must be undertaken at night. DuckieTown can be used for situational awareness for projections and spoofed images in the training of teleoperation operators. They must understand that these attacks can occur and have the ability to confuse the human operator into thinking the autonomous cognition has failed to detect a lane marking.

Scenario: An external threat actor spoofs the road markings to manipulate the drive logic to veer the vehicle off the road.

Attack Sequence:

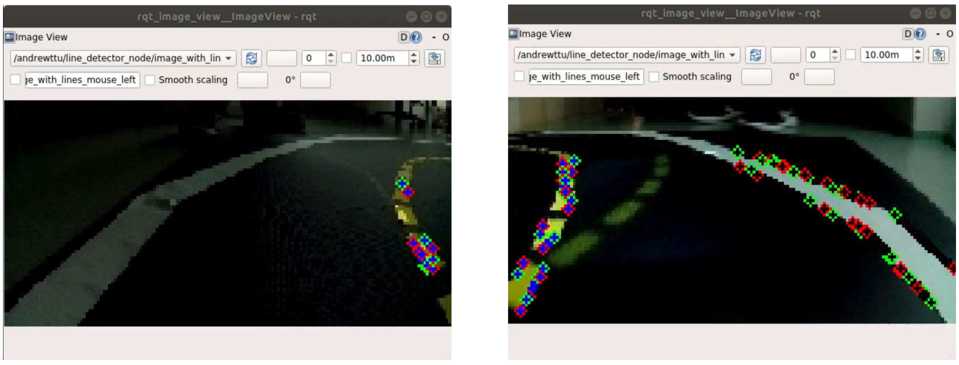
1. Attacker observes the autonomous self-driving vehicle to understand how the autonomous drive cognition makes decisions.

2. Attacker crafts a spoofed image of a lane marking for projection on the driving environment.

3. Attacker positions a drone with a projector attached to it, in proximity to the vehicle and uses a remote control to initiate the projection attack.

Results: The spoofed projection attacks were unable to alter the driving actions of the DuckieBot. Figure 58a shows the faint image of the phantom spoofed yellow line which is barely visible due to the bright profile of the driving environment. Figure 58b visibly shows the spoofed line marking, due to a larger spoofed image being projected by the attacker. The figure 58b image, from the DuckieBot camera shows that the autonomous drive cognition is detecting the edges and texture of the yellow lines and white boundaries but is not detecting the spoofed projection image. This is due to the lack of edges, texture and geometry of the spoofed projection image.

Multiple diverse attack methods were trialled, the spoofed projection images were left projecting on the road surface for 10 minutes, the size of the images were increased, the definition of the images increased, projection on different sections of the floor and different environmental light. The DuckieBot was resilient to the projected road markings attack and the autonomous drive cognition was not manipulated.


rqt_image_view__ImageView - rqt
Image View
/andrewttu/line_detector_node/image_with_lin
0
10.00m
je_with_lines_mouse_left
Smooth scaling
0°
rqt_image_view__ImageView - rqt
Image View
/andrewttu/line_detector_node/image_with_lin
0
10.00m
je_with_lines_mouse_left
Smooth scaling
0°

(a) Low Noise Signature - Projected Lane Marking

(b) High-Noise Signature Projected Lane Marking

Figure 58: Scenario 1 - Projected Road Markings Attack

S02 - Tampered Lane Markings

Problem: Although this threat seems simplistic in the experimental test bed environment, the implications for a real-world operational vehicle are stark. An attacker can use a 3D printer to print a tampered road patch and place it on the road markings of a highway. If this test had occurred on an autonomous vehicle travelling at 40 mph it would have resulted in physical damage to the AV. This attack shows the usefulness of the cyber-physical AV. The cyber-physical AV enabled this attack to be experimented repeatably and we were able to try different lane marking manipulations. This is an improvement on the methods used by Sato *et al*. [249] where they used a simulation for testing and this simulation environment wasn't able to replicate the role of the teleoperation or camera sensing. Through testing this attack in the DuckieTown, we can see that the teleoperation operator must maintain situational awareness of the road environment if there have been any manipulations by a threat actor or environmental damage. In translating this scenario to the real-world AV we were able to detect that the operational vehicle would be susceptible to this same attack. From this experiment, a greater examination of the sensing and detection algorithms of the real-world vehicle was conducted and updates to the multi-sensor fusion were made to mitigate the risk of this attack.

Scenario: An external threat actor tampers with the road markings to manipulate the drive logic to veer the vehicle off the road

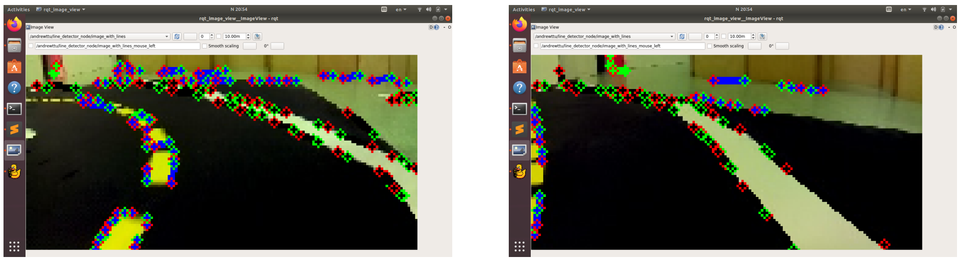
Attack Sequence:

1. Attacker observes the autonomous self-driving vehicle to understand how the autonomous driving cognition makes decisions.

2. Attacker, using the understanding of the drive control algorithm, perturbs the road markings in the DuckieTown environment.

Result: Perturbation of a road marking can manipulate the drive algorithm to cause the autonomous self-driving vehicle to veer off the intended path of travel.

In the first experiment the attacker tampered with the yellow lane markers to manipulate the autonomous self-driving vehicle to drive off the road. The curve road part was changed to a straight trajectory and the angle of the lane borders (white lines) were reduced to lessen the width of the road. As Figure 59b demonstrates, the change to the road markings is demonstrable in the DuckieBot camera sensor footage, from the expected road markings exhibited in Figure 59a. The first experiment was successful in manipulating the autonomous drive cognition of the DuckieBot, however, the DuckieBot autonomy is programmed to firstly respect the lane boundaries. The DuckieBot followed the tampered yellow line until it detected the lane boundary and then adjusted it's travel path to the correct route.



(a) Normal Lane Markings *(b) Manipulated Lane Marking*

Figure 59: Scenario 2 - Tampered Lane Marking Attack

In the second and third the attacker extended the yellow lane markings further into the lane boundaries. The DuckieBot still respected the boundaries and corrected the path of travel.

In the fourth and fifth experiment the attacker removed the lane boundaries and extended the yellow lane markings, as shown in Figure 60 . The attack was successful and the DuckieBot veered off the DuckieTown environment and was unable to recover.

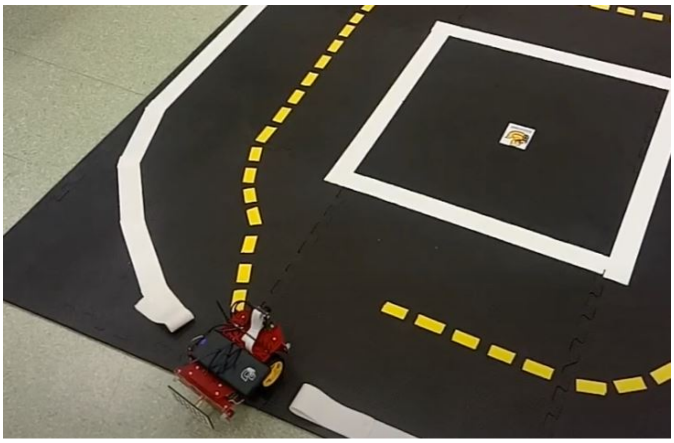


Figure 60: Extended Manipulated Lane Marking

S05 - Firmware update compromise

Problem: Malware in a distributed system provides interesting observations, an autonomous vehicle could lose access to a secure network and connect to a more vulnerable network which would allow malware to propagate more extensively. In testing the findings of Weiss *et al.* [308], we were able, in DuckieTown, use real malware and observe how it propagates in an AV system. We were able to clean the system and repeat the attack to observe any differences in behaviour. In translating the results of the malware attack to the real-world AV shuttle, the AV engineers were unaware of the risks posed in connecting wireless networks in the transit environment. There are many applications for this range scenario for training. Firstly, this scenario would be useful to test incident response to malware in AVs. Secondly, it would be beneficial for the engineers to understand the risks posed by a lack of validation of firmware updates and how malware can spread within a distributed system.

Scenario: An innocent maintenance engineer executes a malicious cryptocurrency or ransomware malware hiding as a firmware update for a vehicle system created by an angry mechanic/insider.

Attack Sequence:

1. Angry Mechanic uploads malware from dark web or publically available repository.
2. Malware script is packaged as a bash script that is labelled "update".
3. Maintenance engineer initiates "update" script with intention of update AV shuttle firmware.

Result: The "update" firmware was executed by the innocent maintenance engineer. The update firmware contained a bash script which executed a cryptomining program. Once infected on a host computer the malware installs several libraries and processes for it's operation and then tries to install zmap (net-work scanner) and ssh pass (utility for establishing SSH connections). It uses zmap, in an infinite loop,to discover then network and find embedded devices with port 22 (SSH) open. If these are found, it connects to the device using ssh with default passwords. It then changes the configuration settings of the device to allow a connection to a command and control node used for cryptomining. On the DuckieBot, the malware installed it's libraries and zmap and ssh pass and began a zmap scan of the network. The DuckieBot was on a private 4G network that also had another AV connected. As these devices do not use default passwords it was unable to establish a connection to them. The zmap scans only marginally impacted the performance of the network of the DuckieBot. The zmap scan was sending 50,000 packets to the target device, but these are only looking for open port 22. An interesting event happened during the experiment. The 4G cellular private network lost connection during the malware execution and the DuckieBot switched over to an open wireless network connection (controlled by us). The zmap process then started to scan the network for open embedded devices. The malware attack was attempted again and this time the wireless campus network was removed. The malware behaved in the same manner and was unsuccessful in brute forcing the DuckieBot.

S06 - Eavesdropping of AV Shuttle operations

Problem: ROS is highly insecure. The version that the DuckieBot is running is the same as the vehicles used by real-world AV Shuttles. There is no authentication and secure communication of the ROS Master. The ROS Master also uses HTTP so it is vulnerable to a number of other malicious web application attacks. The AV testing environment enabled us to test on a real-time system to understand dynamically the information that can be gathered from reading ROS messages and the possibilities of how this information can be used to develop an attack on the vehicle platform. In translation of this to our real-world AV, the mitigating action is to filter the ROS port with a firewall rule. However, if the

attacker gains access to the internal network of the AV system there is little possibility to prevent this attack other than to upgrade to the latest version of ROS, ROS2, which is still under development.
Scenario: An external threat actor eavesdrops on the ROS vehicular messaging system for information gathering.
Attack Sequence: For this attack, the attacker needs to be on the same network as the vehicle.

1. Attacker scans the network and identifies the vehicle

2. Attacker eavesdrops on the ROS communication by spoofing the ROS Master.

Result: The attacker was able to spoof the ROS Master easily and read the ROS messages which are used for AV operations. The attacker was able to generate a ROS graph that showed all of the communication ROS messages (picture not shown/included for anonymity reasons). From this, the attacker could develop a diverse range of attacks such as injection of ROS commands to manipulate a ROS node and replay attacks.

S07 - DDoS Teleoperation Network
Problem: The DDoS attack is an important scenario to replicate in a cyber range due to the loss of control of the teleoperation operator to safely stop the vehicle. This scenario was interesting for the real-world AV shuttle teleoperation staff. When the DDoS attack was conducted the teleoperation console froze and only when the network was re-established did they see that the AV had crashed. This scenario is important for situational awareness training.

Scenario: An external threat actor conducts a denial of service of the short-range wireless network of the autonomous self-driving vehicle.
Attack Sequence:

1. Attacker scans wireless and cellular networks of the vehicle using scanning software such as nmap or airmagnet.

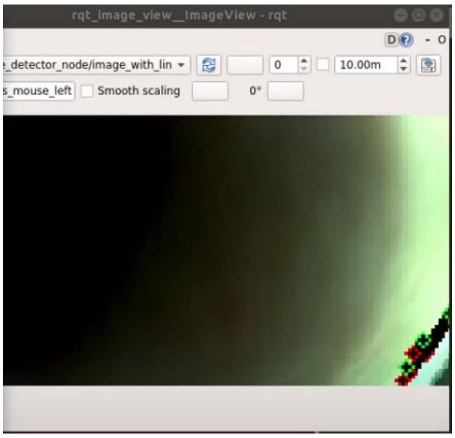
2. Attacker finds the WiFi access point connecting to the human operator console and autonomous self-driving vehicle.

3. Attacker De-authenticates the devices connected to the WiFi access point.

Result: A scan of all wireless networks was conducted on the attackers PC. The attacker used a wireless scanning device that can be considered a malicious access point that acts as a man-in-the-middle between the wireless network and the client device. It can scan, capture traffic and execute a number of attacks such as capturing passwords of insecure network protocols.

The deauthentication attack was attempted twice. Both attempts were successful. Figure 61a shows the teleoperation console after it loses access to the network connection with the DuckieBot and the DuckieBot accelerates off the road. Figure 61b shows the DuckieBot impacting the wall when it loses connectivity. The DuckieBot continues to accelerate on hitting the wall.

S08 - Smoke machine sensor perturbation
Problem: The expert from the autonomous robotics for logistics organisation requested this test case as they wanted to see environmental impacts on the cyber-physical system


rqt_image_view__ImageView - rqt
_detector_node/image_with_lin
0
10.00m
_mouse_left
Smooth scaling
0°
(a) Camera Sensor Vision - DDoS Attack Crash

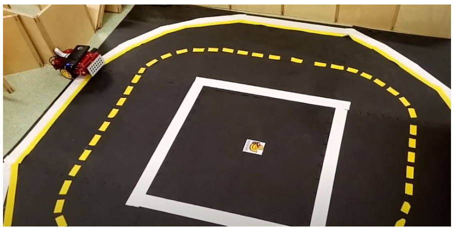

(b) DuckieBot crashed after DDoS

Figure 61: Scenario 7 - DDoS Teleoperation Network

and how they can relate to their real-world autonomous systems. The test case demonstrated the utility of cyber-physical AV test bed in being able to simulate diverse environment conditions. Based on the results of the test case it may be possible to include safety testing in the scope of the test bed.

Scenario: An external threat actor uses a smoke machine to perturb the camera sensor vision and alter the driving course of the vehicle.

Attack Sequence:

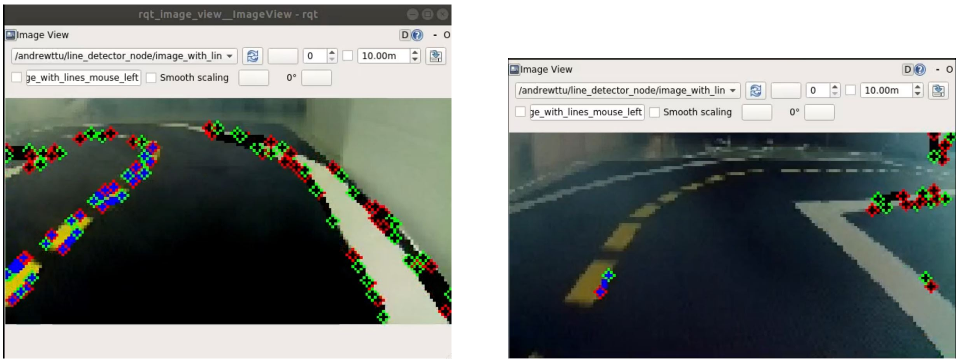
1. A 400w smoke machine is placed next to the environment. The smoke machine is filled with special liquid and then activated using the command controller. Smoke envelops the driving environment.

Result: The experiments were conducted under three different lighting conditions: controlled lights, natural light, controlled dark lighting. In all lighting conditions the smoke was able to perturb the camera sensor to alter the driving path of the DuckieBot to crash out of the road environment.

The initial experimental tests, which were unsuccessful in altering the DuckieBot path, showed that the most important variables were the denseness of the smoke and the ability of the smoke to linger in the air to envelop the camera. The first three smoke experimental tests demonstrated the autonomous driving cognition being lost due to the smoke hazard, however, as the smoke stream was momentary, the detection of the lane markings were recovered in time to navigate accurately. Figure 62a shows the lane detection functioning, and Figure62b shows the smoke perturbing the lane detection of the lane markings.

5.1.5 Discussion

The MIT DuckieTown cyber-physical AV shuttle test bed demonstrated it's use in validating the viability of proof-of-concept attacks such as that of the projector attack and the spoofed lane keeping assistance. The test bed enabled agility and repeatably of testing which facilitated greater understanding of the complexity of implementation of cyber attacks on AVs as well as the challenges for situational awareness for AV operators. A clear representation is the projector attack which demonstrated that it was very difficult for the


rqt_image_view__ImageView - rqt
Image View
/andrewttu/line_detector_node/image_with_lin
0
10.00m
ge_with_lines_mouse_left
Smooth scaling
0°
Image View
/andrewttu/line_detector_node/image_with_lin
0
10.00m
ge_with_lines_mouse_left
Smooth scaling
0°

(a) Lane Detection *(b) Smoke Perturbed Lane Detection*

Figure 62: Scenario 8 - Smoke machine sensor perturbation

adversary to accomplish due to the lighting, projection and camera angle requirements. The WiFi test case provided insights into possibilities for interoperability and human operator research. The vulnerabilities of the network interface, exploited in the cybersecurity test case, impacted the vehicle behaviour and human control.

The results of the testing were provided to the iseAuto, real-world AV Shuttle program. Based on the results analysis, the AV shuttle operator identified a number of vulnerabilities in the AV shuttle architecture. This resulted in the updating of the network package to stop the vehicle in the event of network unavailability or outage. Furthermore, the results helped to educate the teleoperation AV shuttle operators about some of the scenarios they could encounter from an adversarial actor in the driving environment and based on this it assisted in initiating a discussion on what decisions the operator would make when faced with a scenario such as the projection attack or environmental perturbations.

The feedback from the iseAuto concluded that the cyber-physical test bed offered a platform for which they could test corner and edge cases that would be out-of-scope of the real-world vehicle due to cost and risk impacts. It helped the iseAuto AV shuttle program in understanding how their AV Shuttle could be impacted by cyber attacks and with prioritising which attacks were most likely and require further testing on the real AV shuttle.

5.2 ADSecLang - A Domain Specific Language for Cybersecurity Testing of Autonomous Vehicles

Vulnerability testing of AD to cyber attacks is a burgeoning field of research. Initial contributions to this field have focused on novel vulnerability discovery utilising penetration testing methods [92] [33] and fuzzing [154, 298]. However, there exists a gap between this novel, experimental work and the practical implementation of testing to validate the operational readiness of real-world AD systems. Real-world, operational AV testing requires a more rigid approach centered on a structured testing methodology aligned to composite vehicle development and test validation processes. For safety validation testing, domain-specific languages for safety scenario generation, such as SCENIC [81] and ASAM OpenSCENARIO [45], provide a systematic expression that enables a common taxonomy, traceability of testing processes and repeatability and automation of testing for scalability. Yet, there exists a sparsity of research on the development of a domain-specific language for cybersecurity testing of AD systems. One of the primary benefits of the de-

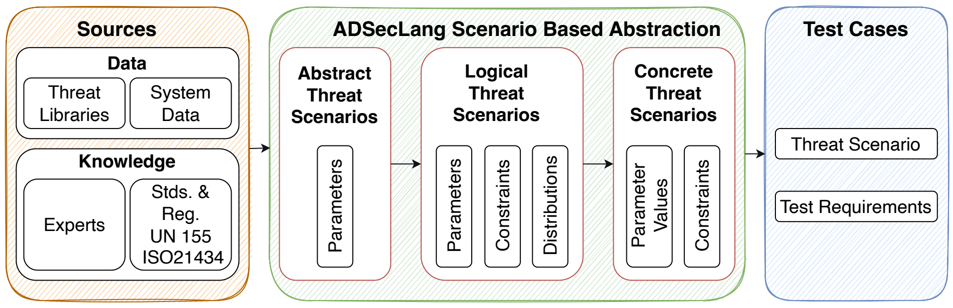


Figure 63: ADSecLang: scenario-based abstraction view.

velopment of a domain-specific language for cybersecurity is that it can simplify the task of writing scenarios for security by providing a concise syntax. In addition, the lack of a domain-specific taxonomy for cybersecurity testing of AD systems further challenges the development and evaluation of AD security testing tools, processes, and methods.

The aim of exploring this problem is to develop such a language, which we call ADSecLang. ADSecLang acts as an intermediary layer in the testing process, which constructs scenarios for cybersecurity through the translation of functional threat descriptions to concrete test cases. Figure 63 depicts the scenario-based abstraction of ADSecLang, which represents the incremental and iterative definition of the threat scenario. First, the abstract description of the threat scenario originates from adversarial analysis, which can leverage sources such as threat libraries, system data, and other knowledge-base repositories. Second, a logical, syntactical expression of the threat scenario is created using a taxonomy. Finally, the technical description of the threat scenario is integrated within the AD simulation testing platform. ADSecLang aims to contribute greater intuition through readable, concise syntax for the development of adversarial agents in simulation testing that would otherwise require complex expressions and constraints. ADSecLang requires the tester to consider all elements of the threat model from attacker tools to desired attack outcomes at both an abstract and parameterised level of abstraction. To demonstrate the utility of ADSecLang, we initially focus on semantic AI security and we evaluate the language to support two use-case scenarios of a camera manipulation attack.

5.2.1 ADSecLang: The Proposed Solution

This section introduces the attack taxonomy used to support the development of ADSecLang and presents the cybersecurity testing framework where ADSecLang can be adopted.

Attack Taxonomy

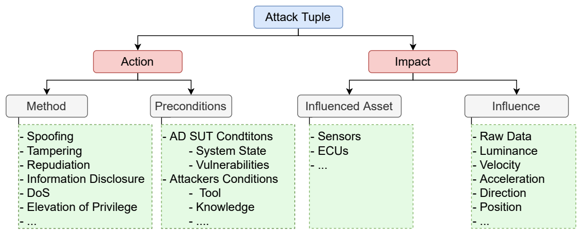


Figure 64: Attack Taxonomy - Detailed Description.

The attack taxonomy of ADSecLang (as shown in Figure 64) categorizes cyber attacks into two domains: **Action** and **Impact**.

Action
represents the execution of an attack method. The success of an action depends on the fulfillment of one or more preconditions. As a result, we subdivide the Action domain into two sub-domains: **Method** and **Preconditions**. The **Method** is defined as the threat technique. This threat description can be derived from a functional description such as STRIDE, Attack Trees, or a textual interpretation. **Preconditions** are a set of conditions that must be met to execute an attack. These preconditions must be inherent attributes that already exist and are not generated by the execution of the attack. The preconditions can be further divided into two categories: conditions on the AD System-Under-Test (SUT) and conditions on the attacker.

- *AD SUT Conditions* are categorized into requirements for the state of the tested system and vulnerabilities within the system. System state conditions refer to the requirement that the target system must be in a specific state (such as a particular version of an operating system, system software/application, or a specific hardware/software state, such as firmware update status) for the attack to be executed. System vulnerabilities refer to exploitable weaknesses in the system's design and operation.

- *Attacker Conditions* can be further segmented into three types: attack tools, attacker knowledge (capabilities, skills), and the level of privileges that an attacker can obtain. The tools and knowledge of the attacker help to profile the type of threat actor capable of conducting the attack. The level of privileges refers to the permissions needed to access or manipulate target system assets. An example would be permission to run processes on the target or existing access to the target asset to manipulate data.

Some attack methods can only be executed successfully when multiple preconditions are met simultaneously. Such conditions will be grouped within braces {}. For example, the precondition `[{A, B}, C]` can be interpreted as 'A and B must be met simultaneously, or C must be met'. To encompass the requirement for multiple preconditions, we define an **Action Group**:

```
action: [method, preconditions]
  method: [category, description]
  preconditions: [precond1, precond2, ...]
    precond1: [category, description]
    precond2: [category, description]
```

Impact
Executing an *Action* will introduce one or more **Impacts** into the system. In other studies [43], these impacts are also denoted as consequence and effect. Although *Impacts* represent the outcomes and effects of attacks, they can also serve as preconditions for subsequent attacks. Consequently, some researchers [164] have alternatively referred to them as post-conditions. In our taxonomy, the utilization of *Impact* aims to identify the direct consequences of an *Action*, which may additionally serve as preconditions for further attacks. The term 'goal' in the attack model represents the ultimate impact. The dimension of *Impact* can be subdivided into two sub-dimensions, namely **Influenced Assets** and **Influence**, which serve to identify the assets directly affected by the *Action* and

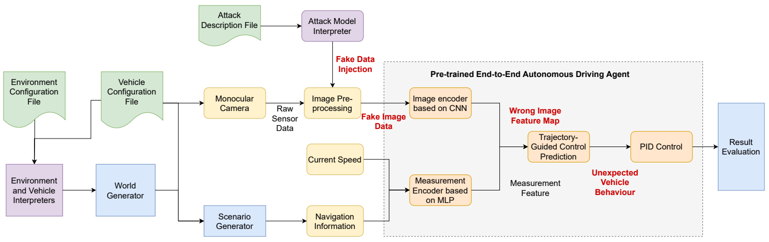


Figure 65: ADSecLang Cybersecurity Testing Workflow - Camera Attack.

ascertain the direct impact incurred on these assets. **Influenced Assets** can be characterized by their respective category and name. For example, the sensor category can include cameras, radars, LiDAR, GPS, or any other AD sensor. The electronic control unit (ECU) category comprises brake control ECUs, engine control ECUs, tire pressure monitoring ECUs, or any other vehicular ECU. **Influence** can be specified as its *Parameter* and *Value*, denoting the specific parameter influenced by the attack and the corresponding altered value, respectively. For instance, if we aim to adjust the brightness of an image captured by a camera, we should specify the parameter as luminance and set its value to 300% (indicating that the brightness has been increased to 300% of the original brightness). To achieve the scalability of ADSecLang, users can add new parameters and a value range in the property configuration file. The **Impact Group** is defined as follows:

```
impact: [influenced_asset, influence]
  influenced_asset: [category, name]
  influence: [parameter, value]
```

5.2.2 Semantics of ADSecLang

The safety scenario domain-specific languages are based on scenario abstraction methodologies such as the Pegasus method [45], which segments three levels of abstraction of the scenario: 1) abstract, 2) logistic, and 3) concrete. For example, an abstract scenario could be described as: '*A malicious actor motivated to cause a safety violation using a laser beam device targeted at a car*'. The logical scenario might be: '*A malicious actor using a laser beam device projecting a luminance of approximately 100 to 300% with a pulse width of 0 to 1*'. Finally, the concrete scenario would specify: '*A malicious actor using a laser beam device projecting 300% luminance with a pulse width of 1*'. Within ADSecLang, the abstract describes the cyber threat scenario according to local parameters. The logical cyber threat scenario extends this description by adding parameter value ranges. Finally, the concrete scenario description contains the set parameter values, which will be utilized as the scenario implementation within the AD simulation testing platform. ADSecLang is designed as an extension of the safety scenario languages [45,81], using the same abstraction method, language semantics and syntax. ADSecLang provides an extension to these areas for cybersecurity.

Compilation of ADSecLang

Compilation of ADSecLang involves three configuration files. Each file contains various user-defined parameters:

- *Environment Configuration File*: This file provides adjustable parameters for scene generation, including town, weather, and traffic density. It also allows users to define constraints on these parameters for scene sampling.

- *Vehicle Configuration File*: This file allows the user to define the parameters of the autonomous vehicle; these include the sensors required, the location of the sensors in the vehicle, the type of sensors, the data to be recorded, and the frequency of recording.

- *Attack Description File*: This file is formatted in the YAML syntax and allows users to define the attack model.

The first two configuration files are relatively simple: the *Environment Configuration File* and the *Vehicle Configuration File*. The environment and vehicle configurations stored in their respective configuration files are read as parameters for generating the driving simulation world and transferred to the world generator. The *Attack Description File* is a more complicated design which has two functions:

- The attack description file is utilized to extract the parameters, which are then translated into concrete code implementation for data processing based on the corresponding attack parameters.

- It is also responsible for connecting the simulation environment, attack code, and autonomous driving system. The attack description file defines the input and output interfaces of the attack code. The input interface connects real-time data captured by sensors in the rendering engine in a simulation environment, such as images captured by camera sensors or status information of ECUs. The output interface sends malicious data generated by attacks to the target AD solution.

Cybersecurity Testing Framework

Architecture

The proposed cybersecurity testing framework has diverse modules for environmental, vehicle, and attack configuration, simulation test, and result evaluation (Figure 65). The functions and roles of these modules are as follows:

- *Environment and Vehicle Interpreter:* Reads the environment and vehicle configuration stored in their respective configuration files as a parameter for generating the world.

- *Attack Model Interpreter:* We read the attack description file as attack parameters. We have defined input and output interfaces for the attack model. The input interface obtains images captured by sensors in the real-time rendering engine and completes the data processing corresponding to the attack parameters read by the interpreter in the specific implementation code of the interface. The specific implementation of the output interface is to send the output of the attack model to the user's chosen autonomous driving solution.

- *World Generator:* Initialise the world based on the environment and entity parameters read by the environment interpreter, including object properties and attribute distribution functions. The world generator randomly samples from the distribution function whenever it is called. By reading the sampling results of the world initializer, a specific world is generated in the real-time rendering engine according to certain steps. The generated world contains at least one vehicle and one camera sensor and exposes the calling interfaces of the vehicle and sensors to the attack model.

- *Scenario generation and result evaluation:* We use a CARLA plugin called CARLA Leaderboard [39] to provide us with scenario generation and evaluation of driving violations. Violation testing includes route completion testing, collision testing, red light running testing, stop running testing, lane crossing testing, proxy blocking testing, and timeout testing.

Cybersecurity Testing Workflow

The overall workflow of the system is shown in Figure 65. The attack target system illustrated here is an end-to-end autonomous driving system based on a monocular camera. The target asset in the vehicle of the attack is the monocular front RGB camera.

The workflow is initiated by storing the predefined environment configuration, object properties, and attack description in configuration files. Execution of the *World Generator* uses the *Environment and Vehicle Interpreter* to read the environment information. Subsequently, each time the scenario is generated, sampling is carried out according to the predetermined process, and the sampling results are converted into the parameterized form we designed and then handed over to the *World Generator*. The *World Generator* first initializes the basic configuration of the real-time rendering engine and creates a specific world in the it, step by step, based on the obtained parameters. Once the world is created, the *Scenario Generator* starts generating test scenarios based on the preset parameters. Subsequently, the *Attack Model Interpreter* retrieves the attack information from the *Attack Description File* and injects the manipulated data to the end-to-end AD system based on the parameters specified by the attack model. Finally, the *Results Evaluation* checks conformity of the AV to safety metrics, as aforementioned, as part of the CARLA Leaderboard [39]. Through conducting multiple iterations of the testing workflow it is possible to evaluate the effectiveness of the attack model.

5.2.3 Evaluation Case Studies

This section examines the use of ADSecLang for supporting the security testing of AV systems. It includes a description of the experimental setup (Sec. 5.2.3) and an analysis of results derived from two attack scenarios (Sec. 5.2.3 and 5.2.3). The goal of the experiments is to assess the ability of ADSecLang to generate attack test cases capable of identifying vulnerabilities in AD systems.

Experimental Setup

The experiments were run on a desktop computer with 12th Gen Intel(R) Core(TM) i3-12100F 4-Core Processor, NVIDIA GeForce GTX 1070Ti GPU, and 16 GB RAM. The use-case scenario testing is conducted on the simulator CARLA 0.9.10. The AD solution tested in the following experiments is a trajectory-guided end-to-end AD solution [315]. This AD solution achieves a new state-of-the-art performance on the CARLA AD Leaderboard [39], in which they rank first in terms of the Driving Score and Infraction Penalty using only a single camera as input. The image captured by the camera has a resolution of 900×256 pixels, and the field of view is maximized at 100 degree.

Attack Case I - Strong Light Exposure Attack

Attack Design

State-of-the-art camera attacks [324] have shown that strong white LED light directed at the camera sensor will result in significantly higher hue values and cause the entire image to be completely white. This results in the camera being unable to capture any visual information. This attack is based on the fact that CMOS/CCD sensors can be interfered with by malicious optical inputs and will produce unrecognizable images. The broken image will further affect the victim AV's decision control. As a result, it will cause uncertainties, which may lead the victim's AV to deviate or emergency brake, both of which can lead to a collision and/or other safety violations. Common methods of attacking camera devices are lasers or LEDs. The Strong Light Exposure Attack interferes with the camera's automatic exposure control. Under laser irradiation, the surface temperature will rise rapidly due to the non-uniform temperature field. Avalanche breakdown of semiconductor materials will cause irreversible damage to optoelectronic devices. Whilst we cannot achieve the physical effects of a targeted light on the camera sensor in a virtual simulator, we can modify the data to simulate the profile of the cyber-physical attack.

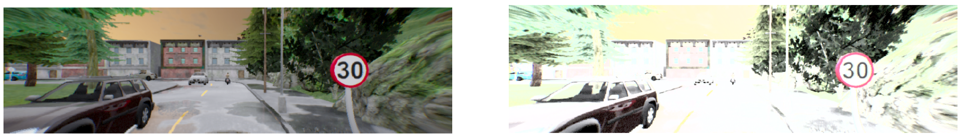
ADSecLang Attack Configuration

The concrete scenario using the ADSecLang attack interpreter file is provided below.

```
attack_name: strong light exposure attack
attack_target: monocular camera-based end-to-end autonomous
    driving system
attack_goal: safety hazard
     action: [method, preconditions]
           method: [tampering, modifying the data captured in
             the asset]
           preconditions: [{precond1 AND precond2 AND
             precond3}]
            precond1: [attacker's knowledge, the attacker
             knows the basic information about the cameras on
              the victim's autonomous driving vehicle]
            precond2: [attack tool, strong LED light]
            precond3: [attacker's capability, attackers can
             shine LED light at AV camera sensor]
     impact: [influenced_asset, influence]
           influenced_asset: [sensor, rgb_camera_front]
           influence: [luminance, 300%]
```

The attack description YAML file is translated using the attack interpreter within the simulation platform.

```
    if(config['attack name']=="Strong light exposure attack"
        ):
    percentage = config['impact']['influence']['luminance']
    file.write('    data = cv2.cvtColor(data, cv2.
        COLOR_RGB2YUV)\n')
    file.write('    h = data.shape[0]\n')
    file.write('    w = data.shape[1]\n')
    file.write('    for i in range(h):\n')
    file.write('        for j in range(w):\n')
    file.write('            y = data[i][j][0]*'+str(float(
        percentage[:-1]) / 100.0)+'\n')
    file.write('            if y > 255:\n')
    file.write('                y = 255\n')
    file.write('            data[i][j][0] = int(y)\n')
    file.write('    data = cv2.cvtColor(data, cv2.
        COLOR_YUV2RGB)\n')
```



(a) Before (b) After

Figure 66: Camera view of attack case 1: before (a) and after (b) the implementation of the Strong Light Exposure Attack.

Table 44: Evaluation result of attack case 1.

Criterion	Result	Value
`RouteCompletionTest`	FAILURE	8.06 %
`OutsideRouteLanesTest`	FAILURE	11.79 %
`CollisionTest`	SUCCESS	0 times
`RunningRedLightTest`	SUCCESS	1 times
`RunningStopTest`	SUCCESS	0 times
`InRouteTest`	SUCCESS	
`AgentBlockedTest`	SUCCESS	
`Timeout`	SUCCESS	

Results

From the comparison of Figure 66a and Figure 66b, we can see that the Strong Light Exposure Attack was successfully implemented. On initiation of the malicious change to the luminance, the monocular camera perception fails to identify the lane lines in the field of view. As a result, the victim AV veered off the lane onto the sidewalk, entering the off-road section of the driving environment. It lost perception and traversed the oncoming lane after being subjected to the Strong Light Exposure Attack. This immediately triggered the failure of the *Outside Route Test* and the *Route Completion Test*, terminating the simulation, as presented in Table 44.

Attack Case II - Laser Beam

Attack Design

Adversarial machine learning (ML), as a form of cyber attack, involves designing a targeted numerical vector to make ML models misjudge and cause system failures and crashes. In this attack test case, the laser construction process is determined by several local-parameters: *intercept*, *injection Angle*, *wavelength*, and *laser width*. This laser attack is executed by randomly selecting a parameter and generating adversarial samples. If the confidence level of the classification is reduced, the current parameter settings are retained, which is often similar to the greedy strategy. After adding a laser beam projection to an image, the image pixels change, which in turn affected the results of the classifier. This adversarial attack, when applied to AD, can target the recognition of traffic lights, speed limit signs, and stop signs. Shining a laser on a stop sign can cause the AD system to fail to identify it correctly, leading to a violation of the required safety condition to stop the vehicle. Also, shining a laser on a traffic light can also create color spoofing attacks. Experimentation with laser beam attacks has shown that if the laser covers the entire traffic light, regardless of its color, the accuracy of detecting red or green lights is hardly affected.

However, if the laser only shines on one traffic light, there will be a significant decrease in the recognition of the traffic light [71]. However, in our testing, we found that if we use this greedy strategy to search for the optimal parameters for 4000 cycles, the generation of adversarial samples is very slow, and it is impossible to inject adversarial samples into the AD test in real time. Therefore, we generate a laser that can make target recognition ineffective and recognise it as another object, by inputting images captured by the camera into an adversarial sample generation program. We then inject this laser in real-time in the AD test scenario. As in the previous case, we assume that the attacker can find appropriate attack scenarios and not be detected by others in advance. For example, the attacker can deploy multiple infrared light sources next to the road where the attacker's vehicle must pass or on a drone.

ADSecLang Configuration

The cyber threat scenario description using the ADSecLang is provided below.

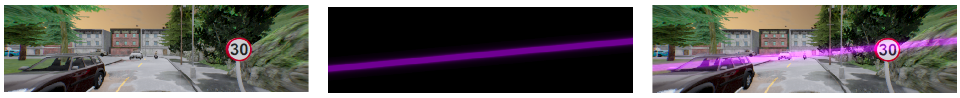
```
attack_name: laser beam attack
attack_target: monocular camera-based end-to-end autonomous
    driving system
attack_goal: safety hazard
   action: [method, preconditions]
    method: [spoofing, shooting laser on the camera]
    preconditions: [{precond1, precond2, precond3}]
     precond1: [attack tool, laser pointer]
     precond2: [attacker's knowledge, machine learning
     adversarial sample generation technology]
     precond3: [attacker's capability, attackers can aim
     lasers at camera sensors on the roadside]
   impact: [influenced asset, influence]
    influenced_asset: [sensor, rgb_camera_front]
    influence: [raw_data, spoofed data]
```

The attack description YAML file is translated using the attack interpreter within the simulation platform.

```
if(config['attack name']=="Laser beam attack"):
file.write('    laser_pattern = cv2.imread("
    laser_for_carriage.png")\n')
file.write('    if laser_pattern is None:\n')
file.write('        print("read image fail!!")\n')
file.write('        return 0\n')
file.write('    laser_pattern = cv2.cvtColor(laser_pattern,
    cv2.COLOR_BGR2RGB)\n')
file.write('    data = data.astype(np.float32)\n')
file.write('    laser_pattern = laser_pattern.astype(np.
    float32)\n')
file.write('    data = cv2.addWeighted(data, 1.0 ,
    laser_pattern, 1.0 , 0)\n')
file.write('    data = np.clip(data, 0.0, 255.0).astype("
    uint8")\n')
```

Results

From the comparison of Figure 67, we can see that the laser beam attack was successfully implemented in the AD simulation. The attack achieved its objective of inducing AV



(a) Before *(b) Generated Laser* *(c) After*

Figure 67: Camera view of test case 2: (a) before the attack, (b) the generated laser beam, and (c) after applying the attack.

Table 45: Evaluation result of attack case 2.

Criterion	Result	Value
`RouteCompletionTest`	FAILURE	71.3 %
`OutsideRouteLanesTest`	SUCCESS	0 %
`CollisionTest`	SUCCESS	0 times
`RunningRedLightTest`	FAILURE	1 times
`RunningStopTest`	SUCCESS	0 times
`InRouteTest`	SUCCESS	
`AgentBlockedTest`	SUCCESS	
`Timeout`	FAILURE	

behaviour to violate a safety condition. As shown in Table 45, the vehicle completed approx. 70% of the route (*Route Completion Test*) and violated a safety condition by driving through a red light (*Running Red Light Test*). The result of the laser attack demonstrated that the laser beam was able to perturb the AD solutions perception of the traffic light, thus causing the victim AV to run a red light.

Future work for the development of ADSecLang will be to extend the language to encompass more diverse semantic cybersecurity scenarios and evaluate the utility of the language to support system-level attack scenarios (Buffer Overflow, Denial-of-Service, Network Attacks, etc.). We further aim to improve the results evaluation module. Metrics for AD testing predominantly focus on safety impacts, however, we would consider it necessary to define metrics that assist in directly evaluating the security of the system under test. Whilst this has proven a difficult challenge, the contemporaneous work on benchmarking for machine learning security and cybersecurity assurance levels (CALs) for automotive systems as conducted by the autonomous vehicle cybersecurity standardisation bodies provides some guidance how to achieve this. We further see the importance of integrating the language within a common AD cybersecurity testing evaluation platform, such as Simutack [79], for an open-source release.

5.2.4 Relation to Existing Work

ADSecLang distinguishes itself from the state-of-the-art as it is the only domain-specific language, to our knowledge, for AD cybersecurity testing and it is designed to integrate within a software simulation testing environment for AD systems. Furthermore, the language has been designed to be agnostic to AD solutions or sensor technology and adaptable to accommodate diverse threat scenarios. SCENIC has been utilized to develop driving scenarios for cybersecurity testing. Salgado et al. [247] used the abstract and concrete scenario composition of SCENIC to create a scenario of a malicious leading vehicle in a convoy to test the robustness of cruise control and collision avoidance. This scenario demonstrates the effect if an attack had already succeeded, whereas the aim of ADSecLang is to incorporate the technical method of attack to assess the performance.

For more conventional threat modeling, *VehicleLang* [145] and *ALLIA (Agnostic Domain Specific Language for Implementing Attacks in an Automotive Use Case)* [312] are the two most prominent studies for legacy automotive architectures. Both of these solutions are focused on modeling cyber threats to connected vehicular systems and focus their case study evaluations on vehicular communication protocols and connected components. *VehicleLang* provides a conceptual contribution, which is the generation of text-based test cases whose feasibility can be validated by expert opinion. *ALLIA* extends this work by providing a technical implementation, which transforms the text-based test case generation into a technical test case implementation.

5.3 SenseFuzz

Fuzz testing of AD software aims to use unsanitised and invalid input to trigger exceptional or abnormal behavior of the driving logic. AD fuzzers are designed in a disparate manner, seeding input from either the sensor data, vehicle dynamics data, scenario and simulator configuration. EnFuzz [48] demonstrated that a collective framework could ensemble diverse fuzzers exhibiting different fuzzing techniques to obtain deeper penetration of one specific type of target, in this instance, application software. As the architecture of AD software relies on a mixture of different sensor technologies and data sources, the innovation required of ensemble fuzzing for AD software is that the framework must be extensible to allow different fuzzers for different targets and target groups. Our idea with this research is to explore such a concept as an ensemble fuzzing framework for AD software and present our ideas on how this could be architected. To this end, we present *FuzzSense* (Figure 68), a conceptual architecture based on a modular, black-box, mutation-based fuzzing framework.

The architecture of FuzzSense is envisioned to integrate within the AD software simulation environment (CARLA, AWSIM, Apollo), allowing diverse fuzzing tools as plug-ins to generate test cases, collect output data in a seed corpus, and mutate new inputs. Our motivation in presenting this work is to provoke discussion within the community on how AD systems are fuzzed, establish community efforts for fuzzing and to gather initial feedback on FuzzSense and understand potential improvements on the foundations of the design of the framework. This work is not a benchmarking study or a statistical evaluation of fuzzing performance, as the motivation is purely to understand how the design of an overarching fuzzing framework for AD software may be achieved. Therefore, to clearly state the aims of this research, we have focused on the development of the initial concept of the AD ensemble fuzzing framework, developed source code, and conducted an initial test case.

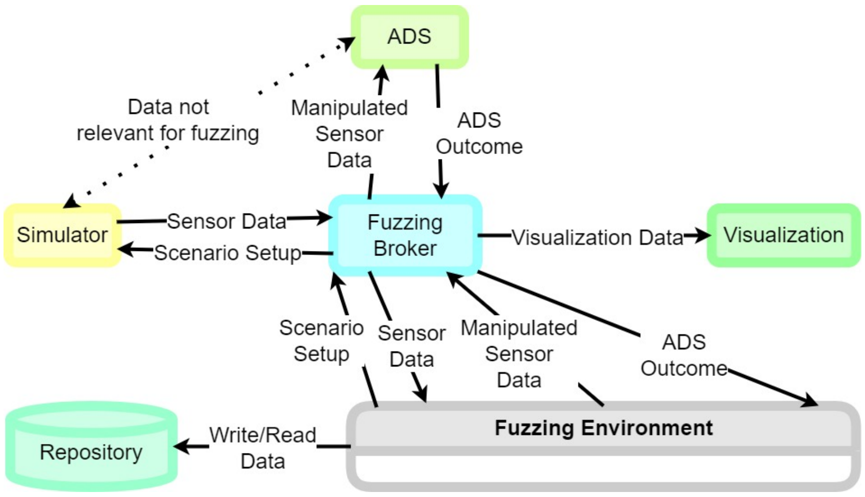


Figure 68: High-level Architecture all Components

5.3.1 FuzzSense

FuzzSense involves the following key components: the fuzzing broker, the fuzzing environment, and the repository. The interactions of these key components with the ADS and simulator are displayed in Figure 69.

Fuzzing Broker The Fuzzing Broker is the central part of the FuzzSense framework, acting as an intermediary layer, facilitating communication between the simulator, ADS, and fuzzing environment. The fuzzing broker has full control over the exchanged sensor data and listens to data, such as steering commands.

While the Fuzzing broker was described as an intermediary for the whole framework, it additionally functions as a controller, initiating and terminating the operations in the connected Simulator and ADS. Depending on the used Fuzzers, Simulator, and ADS, the Fuzzing Broker transforms the sensor data to the required formats of the endpoints.

Fuzzing Environment The Fuzzing Environment is the collection of the components responsible for fuzzing and creating scenarios, manipulating the sensor data, interpreting the results, and mutating parameters. This continues the idea of the modular architecture of the fuzzing framework. It also allows for the decomposition of other integrated modules, as the Mutator is not required to be a part of the fuzzers.

The Fuzzing Environment contains the following modules: orchestrator, mutator, scenario fuzzer, sensor fuzzer/s, and oracle and evaluation.

5.3.1.1 Orchestrator:

The Fuzzing Environment is a composition of diverse components with unique tasks. The role of the orchestrator is to provide a central management function to ensemble these diverse components to achieve the task of fuzzing the selected targets. The idea of a fuzzing orchestrator performing a central management role was inspired by EnFuzz [48], which uses a similar design to integrate and manage diverse fuzzing modules using diverse techniques. The Orchestrator possesses the intelligence in the Fuzzing environment. This is reached by always knowing the current status of the fuzzing campaign and its iterations, therefore, it can start fuzzing iterations, telling each component (Fuzzers, Oracle, Mutators, Fuzzing Broker) when they should perform which of their tasks, monitor the components to understand their status to be able to efficiently

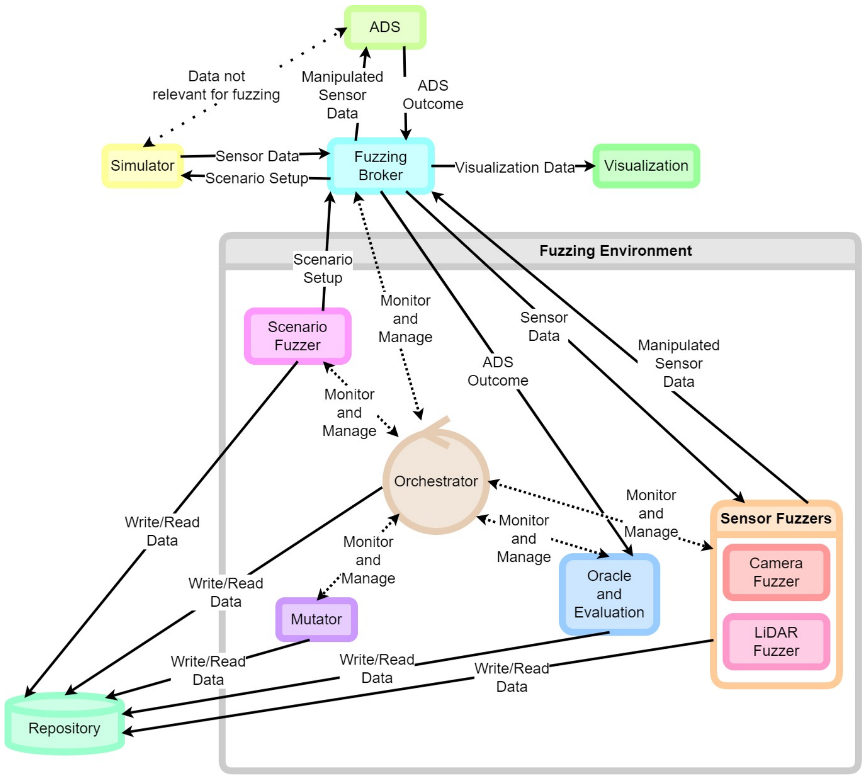


Figure 69: FuzzSense: High-level Architecture of Fuzzing Framework

start the next step with the required components. This requires the Orchestrator to use adapters to communicate to the APIs of the different fuzzing modules. As such, no inter-communication is required for different fuzzing modules; hence, this communication is managed centrally by the Orchestrator. The benefit of central management is that expected new fuzzing modules can be integrated in less time and with less complexity. Further, it even allows decoupling the mutation of parameters and the fuzzers where they are processed.

5.3.1.2 Mutator: The Mutator creates the parameters utilized by the scenario and sensor fuzzing modules. In the first round/s the Mutator is providing the fuzzers with the seeds but does no actual mutation on them. In this architecture, the Mutator is extracted from the scenario and sensor fuzzers. The aim is to allow the combination of different mutation algorithms and fuzzers. Furthermore, it allows a closer synchronization between the mutation of parameters when using multiple fuzzers. For the proof of concept, the mutation is a brute-force/grid search iteration through parameters, where limits are applied and derived from logical boundaries like the perception distance of the sensors.

5.3.1.3 Scenario Fuzzer: Scenario fuzzers use parameters of the driving scenario as the seed pool. These can include weather, pedestrians, and other vehicles. Mutations can be built from the mission, weather, and scenario actors. Prominent scenario fuzzers include only the distinct module creating the scenarios based on parameters given by the Mutator, which is called the Scenario Fuzzer in the FuzzSense architecture.

5.3.1.4 Sensor Data Fuzzers: Autonomous vehicles can use a range of sensor technologies and different hardware and software configurations and can be positioned at different locations on the vehicle. In general, the sensor data of any of those sensors could be fuzzed. The idea motivating our ensemble fuzzing design is that a dedicated sensor data fuzzing plug-in is responsible for each sensor data stream that should be fuzzed. The parameters provided by the Mutator can either be synchronized between several or be mutated individually.

5.3.1.5 Oracle and Evaluation: The Oracle and the Evaluation are giving further intelligence to the Fuzzing Environment. The Oracle and Evaluation component is responsible for creating ground truth, known commonly as the Golden Run. Afterward, every fuzzing iteration must be checked for possible findings, and thus, the Mutator must be provided with an evaluation of the parameters. This framework does not suggest certain conditions once a finding is detected. The idea is to set this based on the subject of testing. For instance, it could be limited to deviations of the trajectory of the Golden Run or only focus on temporal aspects (speed of the vehicle, etc.) introduced by the fuzzing.

Repository In this architecture the repository enables the Fuzzing Environment to write logs, store data and dependent on the communication allow the components to exchange data. When the modules exchange data using the repository, it allows a decoupling and a simpler integration of new components, especially, because the orchestrator is handling the management centrally and thus modules do not need custom integrations with all other required in the Fuzzing Environment.

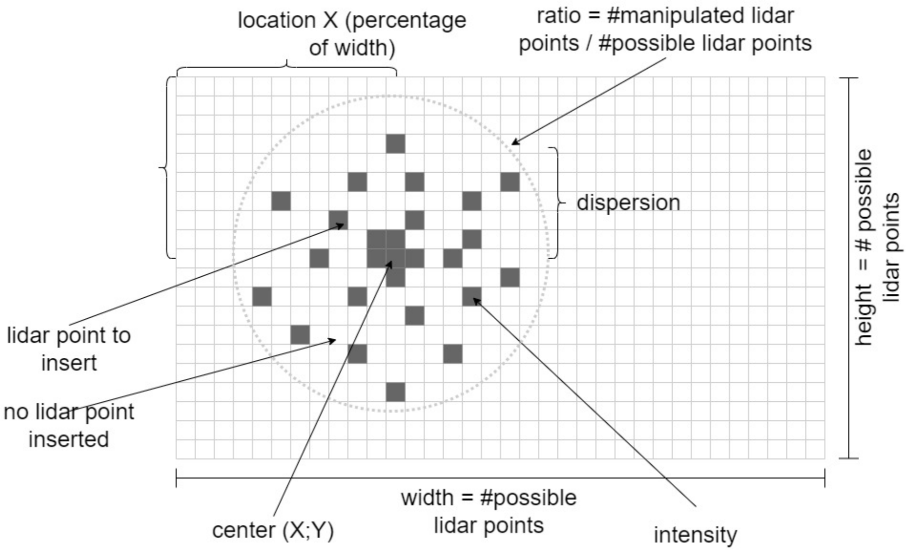
5.3.2 Sensor Data Fuzzing

AD software relies on sensing data for situational awareness and to inform navigation and motion-planning activities. *FuzzSense* is designed to apply manipulations to the sensor data stream before it reaches the downstream AD software. The initial version of the fuzzer manipulates pixels in the camera feed or points in the LiDAR feed. The fuzzer is triggered during a scenario simulation. For each future scenario, the fuzzing test case is mutated based on evaluation of the feedback. The delivery of the manipulation of the sensor data is achieved through the application of changing or adding data in the data stream based on the coordinates given by the *fuzzing mask*.

Fuzzing Mask

The *fuzzing mask* is created based on parameters given by the sensors and vehicle that are to be tested. For the camera stream, which can be represented as a matrix with definitions of each pixel's coordinates, color, and sometimes the alpha channel, the *fuzzing mask* provides a collection of coordinates for pixels that are changed in the camera stream. For LiDAR, the same concept is used to add points to the point cloud, and only the distance is added. Our goal is to achieve several advantages with this approach. First, the same mutation strategy for most parameters can be used. Second, the computation of the next data points to manipulate in the LiDAR data stream is independent of the actual point cloud data. This could potentially increase the performance. Third, by limiting the space of possible manipulations in the search space, possible mutations of the parameters can be drastically reduced to the areas of interest (e.g., in front of the vehicle). Within a point cloud, points can be hidden behind others from the sensors perspective. The concept with the *fuzzing mask* prevents such cases so that no added points are shadowed by other added points (see Figure 70).

The fuzzing mask $\mathscr{F}$ (Algorithm 2) is defined as a set of coordinates where the sensor data is manipulated $\mathscr{F} = \{(x_i, y_i) \mid x_i \in [0, W], y_i \in [0, H]\}$. For the camera sensor, the



location X (percentage of width)
ratio = #manipulated lidar points / #possible lidar points
dispersion
height = # possible lidar points
lidar point to insert
no lidar point inserted
width = #possible lidar points
center (X;Y)
intensity

Figure 70: Fuzzing Mask for LiDAR.

location of the pixel, and for the LiDAR sensor corresponds to the coordinates of a perpendicular plane in the pointcloud where each point is inserted. The third dimension for LiDAR is provided by the distance parameter between the LiDAR sensor and the plane. The coordinates are relative to width, height, and, for 3D data, the center of the plane. For the camera stream, they are taken from the metadata of the sensor stream, and for LiDAR, they are preset and could potentially be mutated.

Algorithm 2 Generate Fuzzing Mask $\mathscr{F}$

```
Require: $r_f, \sigma_f, X, Y, W, H$
0: $(\sigma_x, \sigma_y) \leftarrow (W * \sigma_f, H * \sigma_f)$
0: $r_f \leftarrow W * H * r_f$
0: $\bar{x} = \mathscr{N}(r_f, \sigma_x, X, W)$
0: $\bar{y} = \mathscr{N}(r_f, \sigma_y, Y, H)$
0: for $i \leftarrow 0$ to $r_f - 1$ do
0:     $\mathscr{F} \leftarrow add(x[i], y[i])$
0: end for
0: return $\mathscr{F}$ =0
```

Let r_f represent the fuzzing change ratio, defined as $r_f = \frac{N_c}{W \times H}$. Where: N_c is the number of changed data points, W and H are the width and height of the fuzzing mask matrix in discrete steps (e.g. pixels for the camera stream). The result is expressed as a percentage. Then, let σ_f represent the standard deviation of the manipulated data points, computed as the deviation relative to width W and height H. Together, X and Y are the coordinates of the center of the fuzzing mask and the means of the standard-deviation. $\bar{x}$ and $\bar{y}$ are the vectors corresponding to the each x and y coordinate vector respectively. In line 3-4 W and H ensure, that no coordinates outside of the fuzzing mask are created. Where in line 6 $\mathscr{F}$ is created by column stacking the $\bar{x}$ and $\bar{y}$ arrays with the calculated normal-distributions.

5.3.3 Multi-Stage Approach

FuzzSense combines multiple stages during fuzzing. Each time the fuzzing setup is started, it is called a **Fuzzing Campaign**. Each of the scenarios running with different fuzzing parameters is defined as a **Fuzzing Iteration**. This allows to better distinct between phases and to have an easier understanding of the complete process and architecture. The aim of this process design choice is that the focus for the fuzzing campaign can be chosen with more granularity as the multi-stages allows to provide intelligence to the iterations. The logic when to exit the inner iteration (sensor fuzzing iteration) can be set based on the aim of the fuzzing campaign. This is possible, because the inner and outer iteration (scenario fuzzing iteration) can be logically separated.

Fuzzing Campaign The Fuzzing Campaign defines the whole duration of the fuzzer running. A Fuzzing Campaign consists of one or many Fuzzing Iterations. To start a fuzzing campaign, one or several seeds are required. Each seed contains starting values for each parameter. While there is not any condition met, which qualifies the end of the campaign, new scenario fuzzing iterations are started. The campaign also could be stopped manually. The final step is to stop all required services and store the results from the fuzzing campaign to allow further investigations.

Fuzzing Iteration The Fuzzing Iteration defines one single scenario run. It starts with the parameter mutation and ends once the scenario is stopped because of a failure or because it has successfully finished. The fuzzing of every single data frame is not called iteration. A here defined Fuzzing Iteration includes all those manipulated sensor data frames throughout the whole scenario until it finishes or fails with a finding. As the main focus of the fuzzing is on the sensor data, the mutation for the scenario parameters is not performed in every iteration. Thus, the same scenario is present throughout several iterations. To distinguish also between those two, there can be *Scenario Fuzzing Iterations* and *Sensor Fuzzing Iterations*. One Scenario Fuzzing Iteration consists of one or many Sensor Fuzzing Iterations.

5.3.3.1 Scenario Fuzzing Iteration

The ADS of the AV must act within a scenario to allow relations to its intended real-world use. A scenario defines not only the ego-vehicle itself but also the road, traffic signs, and signals, road conditions, environment, other actors, including their behavior, and the weather conditions. The Scenario Fuzzing Iteration is the outer iteration and contains all Sensor Fuzzing Iterations in the same scenario. It contains the following steps:

Step 1: Mutate Scenario Parameters

Step 2: Create a Scenario and set it up in the simulator and ADS

Step 3: Create Golden Run

Step 4: Start *Sensor Fuzzing Iterations*

5.3.3.2 Sensor Fuzzing Iteration

Within the same Scenario Fuzzing Iteration, the parameters for the Fuzzing Mask should not be the same twice. However, within a new Scenario Fuzzing Iteration, the same parameters can be used again. Each sensor fuzzer takes the original sensor data from the simulator and applies manipulations to the data stream before it reaches the ADS. Those manipulations are single pixels in the camera feed or points in the LiDAR feed. In the current state, within one run, the planned drive of the vehicle, no mutations on the parameters are performed. This means the same fuzzing masks are applied to the data streams from the start to the end of the drive. The mutator is only active between runs. Therefore, compared to a plain simulation, the only computational overhead during a running simulation is the rerouting and manipulation of the sensor data. It contains the following steps:

Step 1: Mutate Sensor Parameters

Step 2: Set scenario up in simulator and ADS

Step 3: Create Fuzzing Masks

Step 4: Start scenario and manipulate sensor data streams

5.3.4 Experiment & Results

Experimental Setup

The evaluation of FuzzSense is conducted in AWSIM, a high-fidelity, digital-twin simulation environment. The target AD system uses the Autoware.Universe software framework. As this instantiation of the AD software uses the LiDAR sensor for perception and localisation, the sensor fuzzing module is configured to fuzz the LiDAR sensor. The evaluation was conducted on a system running Ubuntu 22.04.03 LTS with 1 TB of storage, 32 GB of CPU memory, 10 GB of GPU memory, a 12th Gen Intel® Core™ i7-12700KF processor, and a GeForce RTX 3080 Lite Hash Rate graphics card.

Results & Discussion

The driving scenario consists of a planned navigation in an urban driving environment. We selected an urban environment since attacks can cause more severe effects within a congested operational driving domain. As the vehicle navigates through its planned trajectory, the sensor fuzzing plug-in of FuzzSense initiates its fuzzing mask, manipulating the parameters of the LiDAR 3D geometry. For this set of experiments, the parameters were randomly set at x (0.4),y (0.5), the distance of the fuzzed LiDAR points (30m), and the intensity 0.1. and dispersion (width 100, height 60). The experiments mutated the location and dispersion parameters. The fuzzing broker is fuzzing every frame. In the simulation, the environment exhibits a performance of time of approx. 30 frames per second or 33 milliseconds. Figure 71 displays the initiation of the fuzzing mask (the yellow box is used for identification and does not represent the full mask) to the driving simulation. The fuzzing mask is applied at different distances from the vehicle and different locations within the environment. As shown in Figure 71, the fuzzing mask is located at an approaching distance to the vehicle of approx. 30 meters outside the lane does not produce any unsafe changes in the vehicle the vehicle's behavior.

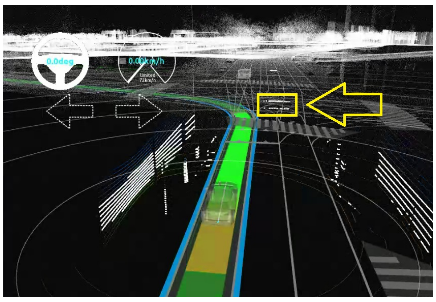


Figure 71: Fuzzing Mask applied to the right edge of lane

Figure 72 displays the movement of the fuzzing mask to a more central location in the driving environment. The fuzzing parameters for amount and dispersion are the same as

Figure 71 in both fuzzing iterations. The parameter for the distance is the same for both. The affect of the fuzzing mask displayed in Figure 72, is that the vehicle detects the fuzzed

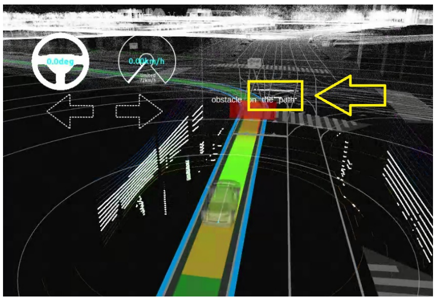


Figure 72: Fuzzing Mask applied to central location of vehicle trajectory

LiDAR points as an obstacle (red wall) and plans a reduction in acceleration to observe the obstacle. This can be seen by the orange color in the planned trajectory.

Figure 73 displays the fuzzing mask applied at a close distance and within the planned trajectory of the vehicle. The vehicle detects the fuzzing mask as an object in immediate proximity to the vehicle and therefore initiates a braking action. The vehicle is unable to recompute an alternative planned trajectory due to the fuzzed points presenting an obstacle across the road and therefore the vehicle is unable to progress.

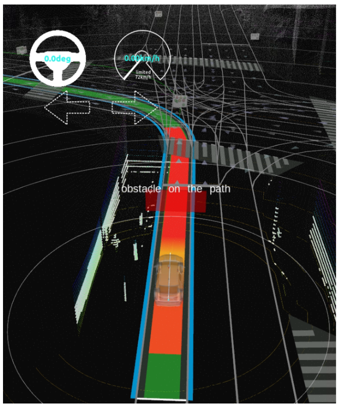


Figure 73: Top down view of vehicle with fuzzing mask affecting planned navigation of the vehicle

The experiments provide initial feedback on the utility of FuzzSense. From observing the behaviour of the AD software, displayed in Figures 72 and 73 we can discern that sensor fuzzing is a useful exercise to find vulnerabilities of the AD software stack. The results indicate that the AD software is either unstable or can be influenced by inserted LiDAR points. We found that when the fuzzing mask was located on or near the planned trajectory of the vehicle, the perception algorithm was unable to filter the manipulated points and instead, observed them as an obstacle. Further to this, when the fuzzing mask was located in close proximity to the vehicle, it resulted in a complete stop of the vehicle.

5.3.5 Related Work

The EnFuzz architecture [48] demonstrates the advantage of combining multiple fuzzers which use diverse techniques of fuzzing, to get a greater and deeper penetration of the target. The EnFuzz design further inspired our adoption within FuzzSense of an orchestrator (monitor) for coordination. Our contribution is unique from EnFuzz as our focus is specific to AD software and we incorporate in the design considerations for the diversity of AD technology and targets.

Aforementioned, there are various fuzzers focused on disparate targets of the AD system. Popularly cited fuzzing tools include DeepRoad [333], DeepTest [281] which target the camera sensor and AV-Fuzzer [170], Auto-Fuzz [339] and DriveFuzz [154] which target the driving scenario. These fuzzers are not designed to operate concurrently with different fuzzers, but focus on a seed pool limited to there target. For the optimization of the search space reduction, these fuzzing tools mainly focus on driving quality and task performance metrics as a measure to direct the mutations towards more promising scenarios where the ego-vehicle is more likely to struggle.

Our work does not aim to compete with these fuzzers nor do we seek to build on there designers. FuzzSense, is an overarching framework whose concept is based on enabling the usage of the fuzzing tools as plug-ins in an integrated fuzzing environment. A future test case would be to use DeepRoad [333] and DriveFuzz [154] within FuzzSense to understand how diverse fuzzing techniques generate bugs.

5.3.6 Future Direction of FuzzSense

Future work, aims to experiment with FuzzSense utlising the modularity to benchmark the performance of different fuzzing plug-ins. Further, advancing the design of the fuzzing mask by adding support for further sensor types. As part of providing FuzzSense open-source, we also aim to actively gather community feedback and develop the framework further.

5.4 ADSecData Platform: Open-Source Data Platform for Autonomous Driving Cybersecurity

AD software must be secure, with decision control optimized to ensure robustness against cyberattacks. A key challenge in achieving this goal is the lack of open-source data specifically for AD cybersecurity. Without available data, software designers do not have an immediate understanding of the considerations for secure design required to ensure robustness against cyber threats. In contrast, there are many open-source datasets for safety validation, algorithm optimization, and sensor configuration. Popular examples include KITTI [84], Waymo [274], Baidu Apolloscape [304], Argoverse [309] and NuScenes [32]. Common datasets for safety validation have enabled platforms such as CARLA Leaderboard [39] to establish challenges to benchmark solutions for perception and trajectory planning algorithms. The problem motivation that this research confronts is that AD cybersecurity doesn't have a readily available source of open datasets available to advance research and there is a lack of guidance on how to conduct cybersecurity research to generate datasets for benchmarking.

To confront this problem, we have developed *ADSecData Platform*, a consolidated platform that provides open-source AD data for cybersecurity. (See Figure. 74), ADSecData Platform consists of a data generation process, which is the method used to generate datasets from simulation and real-world experiments. We validate the platform in a case study using the data generation method to create datasets based on an operational autonomous vehicle (AV) program. We demonstrate the utility of our open-source platform to the community in advancing cybersecurity testing to measure and improve the robustness of autonomous driving systems to cyberattacks.

To construct an AD cybersecurity open-source data platform, we used these guiding

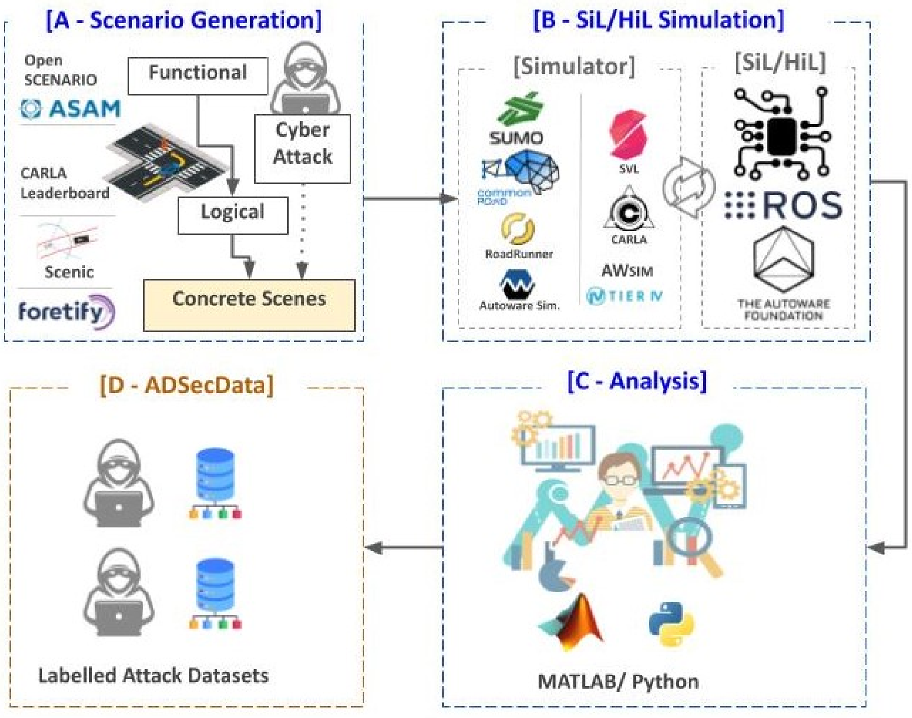


Figure 74: ADSecData Platform - Data Generation Process.

questions to establish an understanding of the relationship of AD data to cybersecurity:

1. *What data types generated by the AD system are utilized for cyber attack test cases?*
2. *What is the utility of each data type to enhancing the cybersecurity of AD?*
3. *What type of metrics are available to benchmark AD algorithms from a cybersecurity perspective and defense mechanisms?*

5.4.1 Autonomous Vehicle Cybersecurity Data

The emerging field of automotive cybersecurity research over the last decade has focused predominantly on the CAN Bus protocol, connected vehicle protocols, electrical and embedded hardware (such as wireless controllers and Bluetooth), and in-vehicle software systems (e.g., infotainment systems). To support the development of defensive technologies and the secure design of communication protocols and software, numerous open-source datasets of automotive telemetry have been created. These datasets primarily address legacy and connected vehicle technologies, with a strong emphasis on the CAN Bus protocol. However, there is a significant lack of open-source cyber attack datasets specific to AD technology. Developing such datasets and promoting the exchange of open-source data are critical steps toward advancing the still-maturing field of AD cybersecurity.

Autonomous Vehicle Data

AD systems generate a vast amount of data from diverse hardware and system components. We classify AD data into four major sub-categories of data sources: *sensing*, *system*, *network*, and *vehicle dynamics*. For each data source, we discuss its value for software development, cybersecurity, and its availability.

Sensing
Sensing data is produced by advanced sensors in the AD system, including LiDAR, cameras, ultrasonic radar, and global navigation systems (GPS, GLONASS, Baidu, Galileo). This data is critical for mapping the driving environment, perception, and localization. However, one of the key challenges with sensor data is the high data rate generated by autonomous vehicles. Xu et al. [322] estimated that diverse sensors could generate approximately 4 terabytes of data per day. The transmission of LiDAR and high-definition camera frames from on-board sensors to edge data logging servers further complicates data collection. Although compression techniques are available to optimize transmission efficiency, there is limited understanding of how these methods impact cybersecurity research in computer vision and perception.

Software Development Value: Sensing data is used by AD software designers to train and optimise algorithms for SLAM, object detection and tracking, sensor fusion and semantic segmentation. One of the many examples of the progress in this area is the CARLA Autonomous Driving Leaderboard [39] which is platform used for the development of AD agents.

Cybersecurity Value: Sensing data can be used to assess vulnerabilities of AD software to adversarial examples and also to generate new attack models for adversarial examples. Select examples include:

- LiDAR point cloud manipulation [35]
- Adversarial examples for camera perception neural networks. [76]
- Light manipulation attacks on camera hardware and driving objects (road signs etc.) [248]
- Fuzzing and parameter manipulation attacks against AD algorithms (Object Detection, Sensor Fusion) [92]
- GPS Spoofing cause uncertainties to trajectory planning algorithms. [136]

Defensive technologies can also be developed from sensing data, these include:

- Kalman filters and ML detection solutions to filter noise from data manipulation attacks. [135]
- Physical intrusion detection solutions which fingerprint patterns of noise from adversarial activity. [228]
- Improvements to the security of ML models to protect against ML evasion, training data poisoning attacks.

Data Availability: Open-Source cybersecurity datasets for sensing, of which there are very few, predominantly focus on camera based perception and neural networks for perception algorithms. Available datasets include:

- *Natural Denoising Diffusion Attack (NDDA) dataset* [252]
- *SlowTrack: Camera based perception latency attack dataset* [181]

System

System data consists of data from the on-board software systems of the AD system. These include the firmware, operating system, application software and real-time operating systems used in the electronic/embedded components such as the electronic control units (ECUs) and micro electronic control units (MCUs).

Software Development Value: System data is used by software developers to debug errors and understand application performance and functionality. Crucial for AV developers is to understand the performance and reliability of the AD software (Autoware, Nvidia Drive, Apollo) and middleware (Robotic Operating System (ROS), Cyber RT).

Cybersecurity Value: System data is used for vulnerability and exploit analysis. Activities that are included in this description include, reverse engineering firmware, code analysis, taint-analysis and fuzz testing.

Data Availability: System datasets are generally available from the manufacturer. These are then used for vulnerability and exploit analysis. Cybersecurity datasets are rare as the responsible disclosure process usually results in the removal and updating of new software. An example of an cybersecurity system artifact are the following:

- *Kia OFFensivE Exploit (KOFFE) metaslpoit module* [82]
- *Mazda Infotainment USB attack* [291]

Network

Network data consists of data produced from the AV internal and external network. CAN Bus is the network of predominance for in-vehicle communication between ECUs which handles critical real-time functions such as braking and steering actuation. Automotive ethernet is gaining in popularity and is mostly used for drive-by-wire communication. Other communication such as MOST is used for infotainment systems and LIN can be found in more upmarket vehicle classes. The difficulty in providing CAN (and most other in-vehicle protocols) datasets is that CAN is used in a proprietary format by vehicle manufacturers. To decipher the meaning of CAN messages, either the manufacturer diagnostic tool is required or knowledge to reverse engineer CAN messages from investigation of firmware and system manuals.

For legacy and connected vehicles great progress has been made and there exist many available datasets and tools to help with the CAN message extraction process [222] . However, to our knowledge there exists no CAN cybersecurity specific datasets for AD technology. Reasons for this could be the enhanced commercial sensitivity of AD technology, more diverse range of AV manufacturers, implementation of encrypted messaging with CAN-FD, cutting-edge nature of AD technology. Other network concepts typical in AD architectures include Vehicle-to-vehicle (v2v) and vehicle-to-everything (v2x) which use wireless and cellular connectivity for connectivity. Different application layer protocols are used for distinct purposes, these may include MQTT for vehicle on-board unit (OBU) to edge communication and Cooperative v2x (C-V2x) protocols that including basic safety messages (BSM) for cooperative perception and intelligent feedback for decision-making.

Cybersecurity research in this field is well-developed and there are many available studies which investigate attack models to the integrity of cooperative vehicular messages and availability of networks which support vehicle data processing and cooperative communication.

Software Development Value: For software developers, network datasets can assist in understanding system interconnection and latency of data flow through situational awareness data to control actions decided by AD software and physical processes made by actuation.

Cybersecurity Value: Network datasets are primarily used for defensive, intrusion detection solutions. Network datasets also aid in developing new attack strategies (DDoS, Replay etc.) and fuzzing strategies to test the robustness of communication architectures. Lately, as more CAN cybersecurity datasets are available, research has focussed on ML and AI solutions for automated attack detection and fuzzing [8]. Within AD architectures, network data is utilised to evaluate the security aspects of cooperative driving such as message trust and authentication. Perhaps the greatest contribution of cybersecurity CAN datasets has been the increase in attention brought by attacks which demonstrate the feasibility of cyber attacks to manipulate safety critical functions such as braking, steering and acceleration. Recognition of these threats has seen the development of security within automotive software architectures (AUTOSAR Adaptive) and new zonal communication architectures for in-vehicle network communications.

Data Availability: Open-Source CAN hacking datasets exist for legacy and connected vehicles, a sample of this long list include:

- *Car-Hacking-Dataset* [258] [268]
- *Survival Analysis Dataset* [104]
- *CAN-Train-And-Test Dataset* [166] [165]
- CANet Dataset [106]
- CrySyS Dataset [83]
- *CIC IoV 2024 Dataset* [213]
- *CAN-MIRGU Dataset** [229]

**The CAN-MIRGU dataset is generated from a vehicle with AD capabilities, however, these capabilities are not detailed due to privacy reasons and the AD functions are deactivated for safety reasons.*

For V2X and V2V selected datasets include:

- *Simulated VANET Attack Dataset* [125]
- *Simulated VANET Attack Dataset* [86]

Vehicle Dynamics

Vehicle dynamics data include body physical movement (lateral and longitudinal pose, yaw etc.), acceleration, braking, steering actuation. Vehicle dynamics is crucial for a software developer and cybersecurity engineer to understand how behaviour at a system-level affects the vehicle. Existing cyber attack research which focuses on vehicle dynamics, predominantly concern themselves with providing artifacts such as docker images of the attack simulation and the code-base for adversarial examples and fuzzing tools. A limitation of this approach is that it requires of custom configuration of the attack in the user environment and an understanding of the vehicle model and metrics engine for data output, used in the original research.

Software Development Value: This data is crucial for control algorithm designers to assess the robustness of control and trajectory planning algorithms. Software developer and control designers will use vehicle dynamics data for backstepping and back-propagation of the AD control software.

Cybersecurity Value: Vehicle dynamics data enables a greater understanding of the affect of cyber attacks to vehicle behavior. The utility of vehicle dynamics data includes research and development of physical intrusion detection systems solutions and root cause analysis.

Data Availability: We are not aware of any datasets for vehicle dynamics in the context of cybersecurity.

Gaps in Autonomous Vehicle Datasets

Our exploration of diverse AD data types and the usage in cybersecurity has identified a number of limitations:

- **Lack of a consolidated research data platform.** Datasets are distributed across github accounts and research papers. There is a lack of consolidation of datasets that would enable security research across the AD technology stack.

- **Siloed research.** Defensive mechanisms are often developed based on a single data type (e.g., CAN, Camera, etc.). The lack of availability of other data sources and an understanding of how this data impacts vehicle dynamics and propagates through the AD system results in the creation of defense mechanisms that lack system-level validation.

- **Lack of cybersecurity data:** There is a lack of data for cybersecurity, and in some of the sub-categories explored, there is, to our knowledge, no data available. The available datasets overwhelmingly consist of legacy and connected vehicles.

5.4.2 ADSecData Platform

In developing a method for generation of cybersecurity data for AD systems, the significant change from legacy vehicles is the focus on vehicle behaviour. As the vehicle is controlled by software and algorithms, it is important to understand the affect to the vehicle from cyber activity and its implications for decision-control. In addition to attacks that directly target AD technologies such as advanced sensors, attacks to network and system components can have downstream affect on autonomous control. The ADSecData Platform (shown in Figure 74) follows a four-stage process for generating data.

Scenario Generation

Scenario-based testing (SBT) involves evaluating the performance of a module or the full AD pipeline (perception, localization, planning, and decision-control) to perform its task during a specified driving scenario. Since the performance of algorithms can vary under diverse scenarios, SBT has become the standardized approach for AD algorithm safety validation and verification testing [116]. Cybersecurity represents an edge and corner case for SBT. For the ADSecData methodology, we propose that scenario generation is a crucial step for cybersecurity, as it is essential to understand whether the effect of a cyber attack on the vehicle differs based on the scenario. Since scenario libraries for AD cybersecurity testing are not available, our methodology recommends using safety validation testing libraries (such as ASAM OpenScenario, etc.) and customizing the scenarios with attack models.

Simulation/Test Environment

As the task of driving can encounter a vast number of diverse scenarios, simulation is the only feasible mechanism to incorporate large-scale testing in an agile manner. Cybersecurity testing should be aligned with safety validation testing, where the choice of test environment is based on evaluating the algorithm's ability to perform tasks. This is part of a testing process that uses regression testing to map scenario test sets from simulation test environments to real-world proving grounds. Within the ADSecData platform, we recommend using low-fidelity test environments for large-scale testing of driving logic, high-fidelity test environments to include testing of advanced sensors (such as LiDAR, Camera, etc.), and real-world proving grounds. Another factor influencing the integrity of cybersecurity data is the tendency of automotive cybersecurity practitioners to provide singular datasets based on attack type. Due to the experimental nature of AD algorithms, sufficient tests need to be run to ensure that anomalous vehicle behavior is caused by cyber activity and not system errors or a lack of optimization of the algorithm.

Another key aspect of the simulation/test environment stage is defining metrics and configuring the format of output data. To quantify the impact of cyber activity on the vehicle, safety metrics and vehicle dynamic parameters are applied. Cybersecurity labels include details such as the initiation of the attack during the scenario, attack parameters (e.g., sensor interference noise level, GPS positioning offset), and their corresponding weighting.

Analysis

The analysis stage involves interrogating the data to assess its integrity and accuracy, ensuring consistency with the experimentation performed. Popular tools, including MATLAB and Python, are used to plot data, visualize patterns, and analyze trends. For example, analyzing a dataset from the trajectory planning module could generate trajectory maps to visualize the vehicle's path and highlight any deviations from the reference path. Analysis

Table 46: Requirements for ADSecData

Category	Requirement
Documentation	• Dataset should be accompanied by general documentation describing content and origin. • Documentation should include description of the attacks in the dataset and how they were executed/recorded. • Documentation should include description of the features (e.g., origin, meaning, range) and their physical context (e.g., how vehicle speed, engine speed and gear are related).
Labels	• Each entry in the dataset may be given a label for identifying whether that entry is benign or an attack.
Parseability, correctness and consistency	• Data should be stored in an appropriate machine/humanreadable format (e.g., PCAP or CSV rather than SQL databases) • All entries should be correctly formatted (e.g., no corrupt entries) • use a single data format for all entries
Age, Size, Objective	• Dataset should not be legacy ($>$ 5 years old etc.) and consist of a balance between benign and cyber attack data.
Completeness	• Dataset should be complete in the sense that no key features or entries have been discarded.
Transformation and anonymization	• Data should not be irreversibly transformed (changing timestamps etc.) and not be anonymised to the point that it bias' detection mechanisms.
Dataset and Attack Realism	• Dataset should include diverse attacks and not be wholly based on synthetic data.

is a crucial activity for identifying problems with the experimentation process and evaluating the quality of the data.

ADSecData

Data should be benchmarked for measurement and comparison. The benchmarks for automotive cybersecurity datasets from Vahidi et al. [296] systematic evaluation of automotive intrusion datasets serve as a good starting point. We utilise their requirements for data in development of the ADSecData Platform and data readiness labels. Table. 46 provide the requirements for ADSecData datasets.

5.4.3 ADSecData Case Study

Target Autonomous Vehicle

The target vehicle is an AV for public transportation, that is an autonomous electric vehicle (AEV). The shuttle operates at Level 4 autonomy (high automation), meaning that it can handle most driving tasks without human intervention in predefined areas, and it is equipped with advanced LiDAR, radar, cameras, and GPS systems to navigate safely and carry out perception tasks in urban environment. Its software backbone is based on ROS and autoware controlling all the driving functionalities and implementing the driving dynamic model of the vehicle.

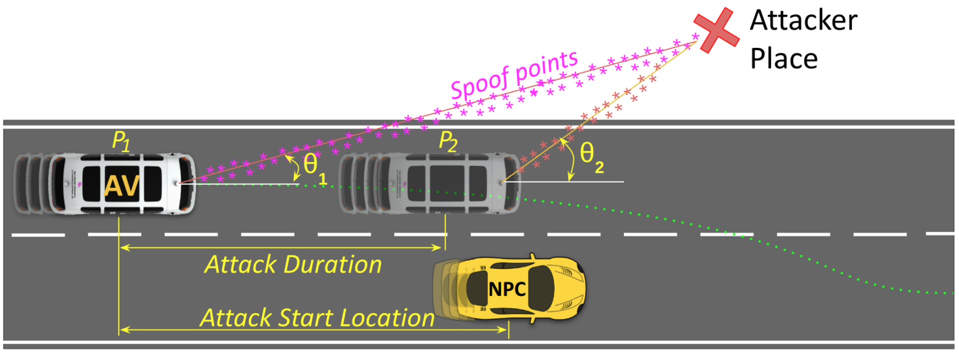


Figure 75: Attack Case 1 Threat Model.

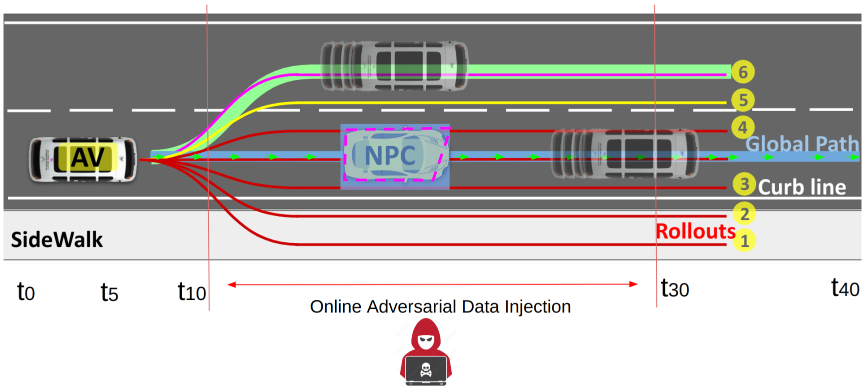


Figure 76: Attack Case 2 & 3 Threat Model.

Scenarios

Our initial dataset consists of 4 attack cases conducted during diverse driving scenarios.

Attack Case 1 - LiDAR point-cloud manipulation: The LiDAR point-cloud manipulation attack, as shown in Figure 75, consists of an adversary with a LiDAR capable of injecting malicious LiDAR point clouds into the LiDARs of the AV. This attack is conducted whilst the AV is attempting an overtaking maneuver.

Attack Case 2 - Position Offset: Attack Case 3 - Message Delay: The attacker creates a spoofed ROS topic which is able to deliver malicious input data of the `Current_Pose` (longitude, latitude, and velocity) to all the nodes of the local planning module. The data manipulation is injected online/dynamically during the critical overtaking manoeuvre involving the AV and NPC (Non-playable character). Figure 76 displays the critical driving scenario and the time frames in which the manipulated `Current_Pose` data is injected into the local planning pipeline cost estimation. The red dashed lines in Figure 76 represent the roll-outs, and the green highlighted, denoting the selected motion-path.

For the manipulation of the `Current_Pose` data, we introduce a deviation to lateral and longitudinal pose. For the lateral pose data, the sensitivity deviation introduced was structured as follows:

- Attack Case 2a: 0.16%
- Attack Case 2b: 0.33%
- Attack Case 2c: 0.5%

This range represents a slight perturbation of pose to a 1m deviation. The longitudinal pose data sensitivity deviation range was structured as follows:

- Attack Case 2d: 0.33%
- Attack Case 2e: 0.66%
- Attack Case 2f: 1.00%

This range is the same as the longitudinal deviation. The difference in percentage comes from the difference in coordinate values of lateral and longitude. The lateral value is almost double those of the longitudinal, and therefore the percentage is doubled.

This attack scenario involves introducing a time-delay into the messages of the Current_Pose topic communicating to the nodes of the local planning module.

We introduced a message delay when the AV passes 2m in front of the vehicle that it is passing in the lateral direction. We introduce 3 different time delays in the message:

- Attack Case 3a: 0.3 seconds
- Attack Case 3b: 0.6 seconds
- Attack Case 3c: 1.0 seconds

The message frequency is approximately 50hz, so this is a message every 20 milliseconds. We chose the above range of deviation of time-delay as it enabled a spectrum of a message from the delay from approximately 15, to 50 messages.

Attack Case 4 - GPS Spoofing: The attack model of GPS spoofing involves an adversary using a transmitter near the AV and interferes with the GPS signals being transmitted.

Simulation/Test Environment

Attack Case 1 was conducted in the high-fidelity CARLA simulator [69]. In this study, we use Carla 0.9.13 as the high-fidelity simulator. Figure 77 illustrates the requirements for the high-fidelity simulator to conduct simulation testing, which are two components, the digital twin of the target AV and the virtual replication of our target environment. These replicated components help us to gain more accurate results of the proposed platform [187]. The AV digital twin is a 3D model of the target real-world world AV shuttle, designed in Blender, a graphical 3d modeling software, and imported and built in Unreal for deployment in CARLA. This model uses the same dimension and sensor configuration (model, position, and orientation) from the real AV shuttle. The environment digital twin, in our case, is identical to the location where the vehicle operates.

This simulation setup was implemented on a desktop computer with the following configuration:

- Intel® Core™ i7-11700K @ 3.60GHz × 16 cores
- NVIDIA GeForce RTX 3080 10 GB
- RAM: 128 GB

Attack Case 2 and 3 were conducted in a low-fidelity simulator. To accelerate the testing, we bypassed the sensing and detection nodes of the algorithm and focused on the planning part by utilizing the low-fidelity simulation feature provided by Autoware.ai and Openplanner. The low-fidelity simulation uses the open-planner 2.5 control algorithm. It provides simulated localization and detection data for the planning nodes and receives the

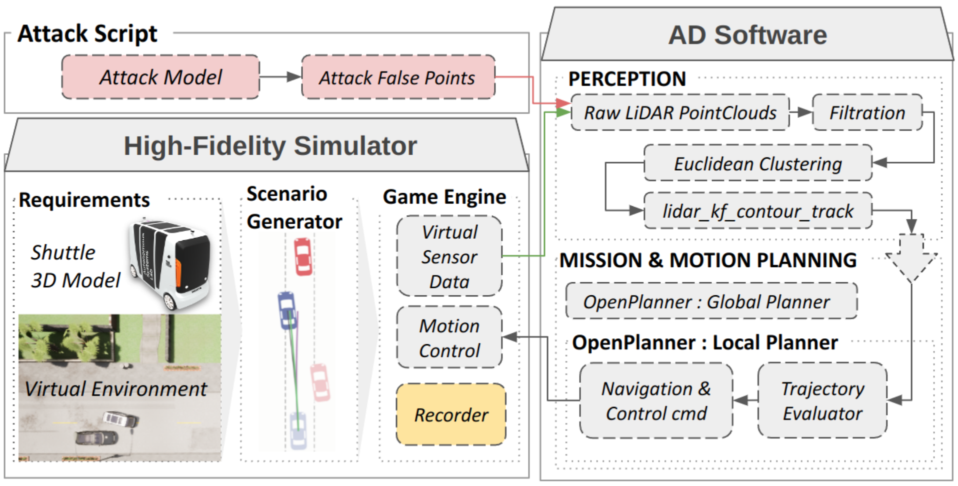


Figure 77: Architecture of the testing platform.

actuation commands to simulate the AV kinematics. This process runs faster due to the low-detail environment required for the simulation and the lack of the process to simulate the sensors.

Attack Case 4 dataset was generated from the real-world vehicle. GPS spoofing activity occurred during a point-in-time of a 3 month trial of AVs in a city in Northern Europe.

Analysis

The data output parameters were defined based on safety, vehicle dynamics and security criteria. A sample of these include, for safety criteria, mission success, violation, break status, distance-to-collision. Vehicle dynamics included steer, yaw, lateral and longitudinal position. Security criteria includes 2 labels, *is_attack* denoting when the attack is occurring and *cyber_weight* which denotes the level of sensor noise manipulation.

ADSecData

The 4 attack case scenarios datasets were generated as a .csv files. Each attack includes a corresponding benign (no attack) dataset to benchmark the stability of the AD algorithms under the given driving scenario. *Attack Case 1* included over 1200 simulations. *Attack Case 2 and 3* included over 900 simulations collectively.

5.5 Discussion

The case study provides a starting point for the development of a common dataset for the community to perform fair and reproducible evaluations of AD algorithms for cybersecurity and defensive mechanisms. The datasets generated from the 4 attack cases demonstrate the importance of following the 4 stage ADSecData method where particular careful consideration is taken in the definition of data output parameters and experimental evaluation analysis. For the development of ADSecData platform, community challenges and a roadmap are fundamental.

Community Challenges

These are the first tranche of community challenges that we recommend for the ADSecData platform:

Ch1 Performance and Accuracy of Semantic Fuzzing Tools

Ch2 Intrusion Detection of Semantic AD Sensor Attacks

Ch3 Robust Sensor Fusion Algorithms

Ch4 Robust and Resilient Trajectory Planning Algorithm

We see these challenges as of most immediate importance and value for the community. Furthermore, we would like to see the community use ADSecData platform to generate a seed corpus for guided semantic data fuzzing tools. As large language models (LLMs) are gaining in popularity, another foreseeable use would be to apply LLMs to ADSecData to generate scenarios for cybersecurity testing. As AD cybersecurity lacks a common scenario library, generation of cybersecurity scenarios would help to close this gap. Finally, IDS solutions for attacks to the AD sensors is essential to mitigate the risk to the AD control. There needs to be more data to understand the profile of cyber attacks comparative to emergency, safety actions from edge and corner cases.

5.6 Future Roadmap of ADSecData

Short term aims of ADSecData platform are to add more datasets from all 4 sub-categories of data types and different vehicle classes and increase the communities awareness of the platform. There will be a need to improve the development of both the front-end and back-end platform to enable secure data sharing and more intuitive user experience. Longer terms aims include a need to investigate metrics for intrusion detection solutions for AD, which is an AI-based system. Traditionally MITRE ATT&CK is used for benchmarking IDS solutions, and MITRE has a framework for AI, MITRE ATLAS. It would be interesting to evaluate how this would work in a practical use-case for AD.

5.7 Relation to Existing Work

There have been attempts by the community to build common infrastructure for AV cybersecurity testing. PASS [115] and Simutack [79] are community simulation testing platforms. Whilst these platforms are valuable to the community and enable accessibility of simulation testing to researchers, the usage of community simulation testing platforms is limited as real-world operators tend to use their own customised platforms. Furthermore, neither of these studies focused on the data aspect of cybersecurity testing as part of their scope. Lauinger et al. [167] developed an attack data generation framework for AVs. Our work enhances this contribution by integrating the concepts of scenario generation and simulation and testing environments for data generation.

From a community data sharing perspective, there are initiatives such as Platform for Innovative use of Vehicle Open Telematics (PIVOT) [226], which is a U.S National Science Foundation project to create a open-source portal for vehicle telemetry data in the context of cybersecurity. However, as of writing this portal was unavailable.

As aforementioned in Section. 5.4.1, there exists a diversity of datasets for legacy and connected vehicles. There are also the studies of Vahidi et al. [296], Lampe & Meng [165] and Lee et al. [169] which evaluate cybersecurity data of legacy and connected vehicles for intrusion detection. However, to our knowledge, there are no existing contributions that focus on the autonomous technology stack of AVs.

5.8 Summary

Within this section, we provide the AD cybersecurity testing community with foundational frameworks for the development of structured and fuzz testing. The ADSecLang framework proposes a methods-based approach to translation of attack models from concept to technical implementation. FuzzSense proposes an Ensemble architecture which aligns with the complexity of the AD software ecosystem as it enable fuzzing of multiple testing targets using diverse techniques. The investigation into the value of data to AD cybersecurity testing uncovered a fundamental sparsity of available datasets for cybersecurity and a lack of knowledge as to the value of datasets and methods for their use to develop defensive mechanisms and offensive toolsets for testing. Datasets are of predominant importance to develop a seed corpus in which to advance more effective test strategies. With each of these contributions we provide a foundational base for the research community to build-upon.

6 Future Direction

AD software is transforming, utilising the advances in AI to control broader areas of the vehicular architecture such as connected interfaces and energy infrastructure (batteries and power management). Further, LLMs extend the capability of AD to enable explainability of the automated driving actions and prediction of events in the driving environment. Future work is directed at developing more robust software architectures to cyber attacks to suit the needs of a complex distributed system environment with a code base of millions lines of code. An architectural approach is necessary as the contemporary focus to the development of defensive mechanisms centers on patching vulnerabilities resultant from cyber attacks. The shortcoming of this approach is presumption that defense will outpace innovation of cyber attacks and due to the safety critical nature of AVs, a successful cyber attack has significant consequences for passenger safety. There are many areas which offer promising research directions:

- Development of software architectures for AVs which is based on security zoning to classify areas as trusted and untrusted. Innovative transformation of in-vehicular architectures to include virtualisation of ECUs for resource sharing and enhanced security configuration and management.
- Development of resilient and robust AD software to protect against semantic level attacks.
- Development of secure protocols for intelligence battery management and power consumption.
- Secure connected protocols for vehicle-to-edge communication to enable resource sharing between on-board and edge compute platforms.

Within the AD testing domain, a greater focus is required on automated testing methods to enable more efficient testing. As stated in the thesis, the lack of standardised methods and tools for attack models and performance benchmarking result in considerable manual effort to reverse-engineer available artifacts for further use. To advance the field, cybersecurity needs to provide open-source, community tools in the same manner as the software development and safety validation community.

7 Conclusion

AD software comprises a complex ecosystem required to support a real-time, safety critical system. AD software must support diverse hardware and technology platforms, integration of mechanical, analog components with digital systems and execution of massive parallel tasks in a time-constrained manner. Whilst software designers are concerned about the robustness of this software to safety validation use-cases, there is a sparsity of research which investigates the design of this software for security. This thesis investigated the design of AD software from the security perspective and focused on 3 critical areas of concern. First, vulnerabilities of AD software to cyber attacks. Second, the affects of cyber attacks to AD software. Third, approaches to cybersecurity testing.

Vulnerabilities of AD software to cyber attacks: AD software is vulnerable to semantic and system-level cyber attacks. The results of the experimental testing demonstrate that malicious data injection, spoofing and jamming attacks on LiDAR, GNSS, sensing data transmission and low-level sensors are successful in the discovery and exploit of vulnerabilities in modular and end-to-end AD software architectures. The modular architecture, exhibited in the real-world case study vehicle, iseAuto, reveals weaknesses of the robustness of its OpenPlanner planning software, NDT-matching based localisation software module and decision-control software modules. A weakness in one software module propagates through the AD pipeline, ultimately affecting the decision-control and result in unsafe driving actions. The results of cyber attacks targeted at the camera perception of the end-to-end architecture of Baidu Apollo demonstrate a lack of training for adversarial examples. The attacks on AWSIM and Baidu Apollo illustrate the limitations of reliance on singular sources of sensor input data. The vulnerabilities discovered in this thesis in the aforementioned software, were reasoned by AD software designers and safety validation engineers as due to a lack of cross validation of input data and mechanisms for resiliency and recovery. he thesis introduces REACT, a proposed architecture for intrusion response in automotive systems. REACT contains methodology for response evaluation, and various response selection methods. We evaluate REACT on 2 diverse attack cases of an adversarial sample targeted at the camera sensor and information disclosure of the infotainment system. The results demonstrates that the LP and SAW algorithms used for optimal selection of response had sub-optimal performance for automated intrusion response in automotive, however, presented encouraging results for proposing follow-up responses to vehicle security operations centre for further action.

Affects of cyber attacks to AD software: One of the aims of this thesis was to develop intuitive methods for security testing that would enable the ability to discern affects to the vehicle from cyber attacks. The thesis developed a method for combined safety and cybersecurity testing which fused the metrics of safety validation which evaluated the vehicles conformance to safety regulations and passenger comfort with attack model parameters. This approach, which was utilised consistently on the real-world vehicle case study, produced valuable insights such as the role of scenario-based testing and temporal aspects in affecting the severity of cyber attack behaviour consequence. The vehicle demonstrated more acute affects to cyber attacks during specific driving maneuvers such as overtaking of passing vehicle and during time periods when the vehicle was attempting the cut-in. Experts reasoned this as being due to engagement of more operations of the software as lane position transitioning and obstacle avoidance are more prominent during these maneuvers. Further, there is a greater need for precision and less tolerance for edge and corner cases. In the thesis' investigation of AD software debugging, vehicle dynamics are added to the evaluation metrics to engage a more intensive analysis of the relationship between cyber attacks, AD software and vehicle behaviour. We found that attacks to the localisation module could be traced to a vehicle dynamic affect, specifically, a GNSS spoofing and jamming attack resulted in an alteration of the vehicles yaw angle and momentum, and its orientation as indicated from the position co-variance and altitude. Furthermore, these attacks triggered the OpenPlanner planning module to execute lane position transitions with greater frequency. The thesis presented the ADAssure method, which involves analysis of the feedback from security testing to develop assertions on the behaviour of the system characteristic with the system being cyber attack. These assertions can then be used for debugging and root-cause analysis. The analysis of low-level sensor attacks, demonstrated how an attack at the system-level, an EMI attack, which altered the values of a steering sensor, could propagate through the sensing and actuation layer through to the high-level control resulting in the sub-modules for the OpenPlanner module, tasked with trajectory generation and waypoint following, generating decisions with the downstream affect of unsafe vehicle behaviour. This analysis showed the benefit of the backstepping technique to pinpoint breakpoints in the software architecture were failures were occurring.

Approaches to cybersecurity testing: Overwhelmingly, cybersecurity testing conducted by the research community uses off-the-shelf, open-source software which is not optimised to the driving maneuvers and operational environments for which it is tested. One of the primary innovations of this thesis is the development and usage of a testing tool-chain approach which utilised digital-twins containing the technology stack of a real-world vehicle. The testing tool-chain approach was used to conduct agile and repeatable testing and regress test cases from a simulation environment to the real-world, physical vehicle. In the cyber test range evaluation, we explored the capability of a small factor testbed to support cybersecurity testing. We found that the small factor testbed could provide insights into the vulnerability of the AD software to semantic-level attacks such as adversarial examples targeted at the camera perception and system-level attacks in the case of a network DDoS on the teleoperation protocol and a SSH brute force attack. These diverse environments, digital-twin simulation, cyber-physical small factor testbed and real-world vehicle, can be utilised to regress testing, with the simulation and small factor testbed offering the benefits of agile and repeatable testing at minimal cost and effort. Another limitation of the research community is the lack of knowledge as to the development of attack models. With our contributions, ADSecLang and FuzzSense, we provide foundational frameworks for the development of community-driven structured adversarial testing and fuzz testing. ADSecLang contributes a method for the translation of attack models from concept to technical implementation. FuzzSense contributes a conceptual framework based on ensemble fuzzing, a modular approach where diverse testing targets and diverse fuzzers can be utilised to gain a deeper penetration of the system. With both these contributions we presented initial results which demonstrated that these tools could be used to find vulnerabilities in the Baidu Apollo and AWSIM software frameworks. ADSecData Platform further provides an initial contribution to enhancing testing methods and tools through the collation and provision of AD cybersecurity datasets. The analysis contained in this thesis found insufficient awareness of the community of the importance of datasets and how data can be used to enhance testing tools, defensive mechanisms and guide efficient testing methods. ADSecData Platform provides a 4 phase data generation process to generate datasets from testing. The initial ADSecData Platform provides datasets and challenges for community participation.

Significance of thesis findings: The main contribution of this thesis is to study the design of AD software within the perspective of cyber attacks. We investigated this:

- via development of diverse attack models utilising a testing tool-chain to discover vulnerabilities in software used in a real-world, operational vehicle.

- via creation of methods able to characterise the affects of cyber attacks to the software and vehicular system.
- via debugging and root-cause analysis of cyber attacks to pinpoint vulnerable areas of the software architecture and analysis of incident response capability.
- via development of platforms and toolsets for structured and fuzz testing.

We provide these contributions in the backdrop of a community-wide effort to ensure the robustness and reliability of AVs to cyber threats. This thesis provides tangible artifacts which include the ADSecLang & FuzzSense code and the datasets from the experiments as collated by the ADSecData Platform.

References

[1] AWSIM. `https://tier4.github.io/AWSIM/` - Accessed: 2024-01-06.

[2] Robot Operating System 2 - ROS 2 (Humble). `https://docs.ros.org/en/humble/index.html` - Accessed: 2024-01-06.

[3] Euro NCAP Working Group on Automated Driving. Euro ncap's first step to assess automated driving systems. Technical report, European New Car Assessment Programme, 2019.

[4] T. 4. Autoware foundation. *The Autoware Foundation*, 2021. Accessed: 2024-03-15.

[5] Y. E. A. Zolfi, M. Kravchik and A. Shabtai. The translucent patch: A physical and universal attack on object detectors. In *CVPR*, 2021.

[6] Y. Abeysirigoonawardena, F. Shkurti, and G. Dudek. Generating adversarial driving scenarios in high-fidelity simulators. In *2019 International Conference on Robotics and Automation (ICRA)*, page 8271–8277. IEEE Press, 2019.

[7] A.Davies. Tesla's latest autopilot death looks just like a prior crash. *WIRED*, 2019.

[8] N. Alkhatib, M. Mushtaq, H. Ghauch, and J.-L. Danger. Can-bert do it? controller area network intrusion detection system based on bert language model. In *2022 IEEE/ACS 19th International Conference on Computer Systems and Applications (AICCSA)*, pages 1–8, 2022.

[9] F. Alrefaei, A. Alzahrani, H. Song, and S. Alrefaei. A survey on the jamming and spoofing attacks on the unmanned aerial vehicle networks. In *2022 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, pages 1–7. IEEE, 2022.

[10] R. Altawy and A. M. Youssef. Security, privacy, and safety aspects of civilian drones: A survey. *ACM Trans. Cyber-Phys. Syst.*, 1(2), nov 2016.

[11] A.Marshall. Uber's self-driving car just killed somebody. now what? *WIRED*, 2018.

[12] N. B. Anuar, M. Papadaki, S. Furnell, and N. Clarke. A Response Strategy Model for Intrusion Response Systems. In D. Gritzalis, S. Furnell, and M. Theoharidou, editors, *Information Security and Privacy Research*, pages 573–578, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[13] S. Anwar, J. Mohamad Zain, M. F. Zolkipli, Z. Inayat, S. Khan, B. Anthony, and V. Chang. From Intrusion Detection to an Intrusion Response System: Fundamentals, Requirements, and Future Directions. *Algorithms*, 10(2), 2017.

[14] S. Anwar, J. M. Zain, M. F. Zolkipli, Z. Inayat, A. N. Jabir, and J. B. Odili. Response option for attacks detected by intrusion detection system. In *2015 4th International Conference on Software Engineering and Computer Systems (ICSECS)*, pages 195–200, 2015.

[15] ApolloAuto. Apollo, 2025. Accessed: 2024-03-17.

[16] Aripaev. Taltech is developing a av together with u.s university. *Aripaev*, 2019. Accessed: 2020-02-15.

[17] V. Au. Mainland china tech giant baidu applies to launch autonomous vehicle trials in hong kong. `https://www.scmp.com/news/hong-kong/transport/article/3286293/mainland-china-tech-giant-baidu-applies-launch-autonomous-vehicle-trials-hong-`Nov. 2024. Accessed: 2025-04-05.

[18] Autolab. Av smart testbed, 2025. Accessed: 2024-03-17.

[19] AUTOSAR. Specification of intrusion detection system protocol. Technical report, AUTOSAR Consortium, 2020.

[20] AVL. Avl zalazone proving ground, 2025. Accessed: 2024-03-17.

[21] J. Axelsson, A. Kobetski, Z. Ni, S. Zhang, and E. Johansson. Moped: A mobile open platform for experimental design of cyber-physical systems. In *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 423–430, 2014.

[22] L. Baresi and D. A. Tamburri. Architecting artificial intelligence for autonomous cars: The openpilot framework. In B. Tekinerdogan, C. Trubiani, C. Tibermacine, P. Scandurra, and C. E. Cuesta, editors, *Software Architecture*, pages 189–204, Cham, 2023. Springer Nature Switzerland.

[23] V. S. Barletta, D. Caivano, M. D. Vincentiis, A. Ragone, M. Scalera, and M. Á. S. Martín. V-soc4as: A vehicle-soc for improving automotive security. *Algorithms*, 16(2):112, 2023.

[24] M. Bashendy, A. Tantawy, and A. Erradi. Intrusion response systems for cyber-physical systems: A comprehensive survey. *Comput. Secur.*, 124(C), jan 2023.

[25] M. Bashendy, A. Tantawy, and A. Erradi. Intrusion response systems for cyber-physical systems: A comprehensive survey. *Computers & Security*, 124:102984, 2023.

[26] R. K. Bhadani, J. Sprinkle, and M. Bunting. The cat vehicle testbed: A simulator with hardware in the loop for autonomous vehicle applications. In *SCAV@CPSWeek*, 2018.

[27] A. Boloor, K. Garimella, X. He, C. Gill, Y. Vorobeychik, and X. Zhang. Attacking vision-based perception in end-to-end autonomous driving models. *Journal of Systems Architecture*, 110, 2020.

[28] Bosch. Facts and figures about electronics and software in vehicles. *Automotive World*, July, 2021.

[29] T. Bouyahia, N. Cuppens-Boulahia, F. Cuppens, and F. Autrel. Multi-Criteria Recommender Approach for Supporting Intrusion Response System. In F. Cuppens, L. Wang, N. Cuppens-Boulahia, N. Tawbi, and J. Garcia-Alfaro, editors, *Foundations and Practice of Security*, pages 51–67, Cham, 2017. Springer International Publishing.

[30] C.-V. Briciu, I. Filip, and F. Heininger. A new trend in automotive software: Autosar concept. In *2013 IEEE 8th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, pages 251–256, 2013.

[31] A. Buscemi, I. Turcanu, G. Castignani, A. Panchenko, T. Engel, and K. G. Shin. A survey on controller area network reverse engineering. *IEEE Communications Surveys and Tutorials*, 25(3):1445–1481, 2023.

[32] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020.

[33] Y. Cao, S. H. Bhupathiraju, P. Naghavi, T. Sugawara, Z. M. Mao, and S. Rampazzi. You can't see me: Physical removal attacks on lidar-based autonomous vehicles driving frameworks. In *Proceedings of the 32nd USENIX Conference on Security Symposium*, 2023.

[34] Y. Cao, C. Xiao, A. Anandkumar, D. Xu, and M. Pavone. Advdo: Realistic adversarial attacks for trajectory prediction. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part V*, page 36–52, Berlin, Heidelberg, 2022. Springer-Verlag.

[35] Y. Cao, C. Xiao, B. Cyr, Y. Zhou, W. Park, S. Rampazzi, Q. A. Chen, K. Fu, and Z. M. Mao. Adversarial sensor attack on lidar-based perception in autonomous driving. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 2267–2281, New York, NY, USA, 2019. Association for Computing Machinery.

[36] Y. Cao, D. Xu, X. Weng, Z. Mao, A. Anandkumar, C. Xiao, and M. Pavone. Generalized decision transformer for offline heterogeneous multi-agent reinforcement learning, 2022. arXiv preprint arXiv:2208.00094.

[37] Y. Cao, D. Xu, X. Weng, Z. Mao, A. Anandkumar, C. Xiao, and M. Pavone. Robust trajectory prediction against adversarial attacks. In K. Liu, D. Kulic, and J. Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 128–137. PMLR, 14–18 Dec 2023.

[38] V. Cardellini, E. Casalicchio, S. Iannucci, M. Lucantonio, S. Mittal, D. Panigrahi, and A. Silvi. An Intrusion Response System utilizing Deep Q-Networks and System Partitions. `https://arxiv.org/abs/2202.08182`, 2022.

[39] CARLA. Carla autonomous driving leaderboard. `https://leaderboard.carla.org/`, 2024. Accessed: 2024-04-16.

[40] CARLA Simulation Project. Epic automotive and carla. `https://carla.readthedocs.io/en/0.9.9/tuto_A_epic_automotive_materials/`, 2022. Accessed: 2022-01-11.

[41] K. K.-C. Chang, X. Liu, C.-W. Lin, C. Huang, and Q. Zhu. A safety-guaranteed framework for neural-network-based planners in connected vehicles under communication disturbance. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2023.

[42] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno. Comprehensive experimental analyses of automotive attack surfaces. In *20th USENIX security symposium (USENIX Security 11)*, 2011.

[43] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno. Comprehensive experimental analyses of automotive attack surfaces. In *Proceedings of the 20th USENIX Conference on Security*, SEC'11, page 6, USA, 2011. USENIX Association.

[44] B. Chen, X. Chen, Q. Wu, and L. Li. Adversarial evaluation of autonomous vehicles in lane-change scenarios. *IEEE Transactions on Intelligent Transportation Systems*, 23:10333–10342, 2020.

[45] H. Chen, H. Ren, R. Li, G. Yang, and S. Ma. Generating autonomous driving test scenarios based on openscenario. In *2022 9th International Conference on Dependable Systems and Their Applications (DSA)*, 2022.

[46] S.-T. Chen, C. Cornelius, J. Martin, and D. H. P. Chau. Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector. In M. Berlingerio, F. Bonchi, T. Gärtner, N. Hurley, and G. Ifrim, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 52–68, Cham, 2019. Springer International Publishing.

[47] X. Chen, C. Fu, F. Zheng, Y. Zhao, H. Li, P. Luo, and G.-J. Qi. A Unified Multi-Scenario Attacking Network for Visual Object Tracking. *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Vir*, pages 1097–1104, 2021.

[48] Y. Chen, Y. Jiang, F. Ma, J. Liang, M. Wang, C. Zhou, X. Jiao, and Z. Su. {EnFuzz}: Ensemble fuzzing with seed synchronization among diverse fuzzers. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1967–1983, 2019.

[49] R. Chevalier, D. Plaquin, C. Dalton, and G. Hiet. Survivor: A Fine-Grained Intrusion Response and Recovery Approach for Commodity Operating Systems. In *Proceedings of the 35th Annual Computer Security Applications Conference*, ACSAC '19, page 762–775, New York, NY, USA, 2019. Association for Computing Machinery.

[50] K.-T. Cho and K. G. Shin. Fingerprinting electronic control units for vehicle intrusion detection. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 911–927, 2016.

[51] H. Choi, W.-C. Lee, Y. Aafer, F. Fei, Z. Tu, X. Zhang, D. Xu, and X. Deng. Detecting attacks against robotic vehicles: A control invariant approach. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 801–816, New York, NY, USA, 2018. Association for Computing Machinery.

[52] L. Claussmann, M. Revilloud, D. Gruyer, and S. Glaser. A review of motion planning for highway autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 21(5):1826–1848, 2020.

[53] G. Costantino and I. Matteucci. Reversing kia motors head unit to discover and exploit software vulnerabilities. *Journal of Computer Virology and Hacking Techniques*, 19(1):33–49, 2023.

[54] J. Cui, L. S. Liew, G. Sabaliauskaite, and F. Zhou. A review on safety failures, security attacks, and available countermeasures for autonomous vehicles. *Ad Hoc Networks*, 90:101823, 2019.

[55] C. . C.Valasek. Remote exploitation of an unaltered passenger vehicle. 2015.

[56] H. Darweesh, E. Takeuchi, and K. Takeda. Openplanner 2.0: The portable open source planner for autonomous driving applications. In *2021 IEEE Intelligent Vehicles Symposium Workshops (IV Workshops)*, pages 313–318, 2021.

[57] H. Darweesh, E. Takeuchi, and K. Takeda. Openplanner 2.0: The portable open source planner for autonomous driving applications. In *2021 IEEE Intelligent Vehicles Symposium Workshops (IV Workshops)*, pages 313–318, 2021.

[58] H. Darweesh, E. Takeuchi, K. Takeda, Y. Ninomiya, A. Sujiwo, L. Y. M. Saiki, N. Akai, T. Tomizawa, and S. Kato. Open source integrated planner for autonomous navigation in highly dynamic environments. *J. Robotics Mechatronics*, 29:668–684, 2017.

[59] S. Dasgupta, A. Ahmed, M. Rahman, and T. N. Bandi. Unveiling the stealthy threat: Analyzing slow drift gps spoofing attacks for autonomous vehicles in urban environments and enabling the resilience, 2024.

[60] P. Dash, M. Karimibiuki, and K. Pattabiraman. Stealthy attacks against robotic vehicles protected by control-based intrusion detection techniques. *Digital Threats*, 2(1), jan 2021.

[61] D. Davidson, H. Wu, R. Jellinek, V. Singh, and T. Ristenpart. Controlling UAVs with sensor input spoofing attacks. In *10th USENIX Workshop on Offensive Technologies (WOOT 16)*, Austin, TX, Aug. 2016. USENIX Association.

[62] M. De Vincenzi, G. Costantino, I. Matteucci, F. Fenzl, C. Plappert, R. Rieke, and D. Zelle. A systematic review on security attacks and countermeasures in automotive ethernet. *ACM Comput. Surv.*, 56(6), Jan. 2024.

[63] N. DeMarinis, S. Tellex, V. P. Kemerlis, G. Konidaris, and R. Fonseca. Scanning the internet for ros: A view of security in robotics research. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8514–8521, 2019.

[64] G. Deng, G. Xu, Y. Zhou, T. Zhang, and Y. Liu. On the (in)security of secure ros2. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, page 739–753, New York, NY, USA, 2022. Association for Computing Machinery.

[65] L. Ding, Y. Wang, K. Yuan, M. Jiang, P. Wang, H. Huang, and Z. J. Wang. Towards universal physical attacks on single object tracking. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(2):1236–1245, May 2021.

[66] W. Ding, I. Alrashdi, H. Hawash, and M. Abdel-Basset. Deepsecdrive: An explainable deep learning framework for real-time detection of cyberattack in in-vehicle networks. *Information Sciences*, 658:120057, 2024.

[67] E. dos Santos and D. Schoop. Towards a simulation-based framework for the security testing of autonomous vehicles. In *6th Embedded Security in Cars USA*, page 15, June 2018.

[68] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In S. Levine, V. Vanhoucke, and K. Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 1–16. PMLR, 13–15 Nov 2017.

[69] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[70] D.Tian. Deep learning, self driving robotic car on a shoestring budget. 2019.

[71] R. Duan, X. Mao, A. K. Qin, Y. Chen, S. Ye, Y. He, and Y. Yang. Adversarial laser beam: Effective physical-world attack to dnns in a blink. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16057–16066, Los Alamitos, CA, USA, jun 2021. IEEE Computer Society.

[72] Z. El-Rewini, K. Sadatsharan, D. F. Selvaraj, S. J. Plathottam, and P. Ranganathan. Cybersecurity challenges in vehicular communications. *Vehicular Communications*, 23:100214, 2020.

[73] Electrive. Zf to test level 4 autonomous system in germany. `https://www.electrive.com/2025/03/11/zf-to-test-level-4-autonomous-system-in-germany/`, Mar. 2025. Accessed: 2025-01-12.

[74] E. S. I. B. ESIB. Accident, loss of control with airbus a320-214 near tallinn airport on 28.02.2018. *Safety Investigations. Investigation report ESIB: A2802118 EECAIRS: EE0180. PDF document.*, 2019.

[75] European Union Agency for Cybersecurity. Enisa good practices for the security of smart cars. Technical report, European Union Agency for Cybersecurity (ENISA), Greece, 2019.

[76] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song. Robust physical-world attacks on deep learning visual classification. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1625–1634, 2018.

[77] Y. Feng, S. E. Huang, W. Wong, Q. A. Chen, Z. M. Mao, and H. X. Liu. On the cybersecurity of traffic signal control system with connected vehicles. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–13, 2022.

[78] B. A. Fessi, S. BenAbdallah, M. Hamdi, and N. Boudriga. A new genetic algorithm approach for intrusion response system in computer networks. In *2009 IEEE Symposium on Computers and Communications*, pages 342–347, 2009.

[79] A. Finkenzeller, A. Mathur, J. Lauinger, M. Hamad, and S. Steinhorst. Simutack - an attack simulation framework for connected and autonomous vehicles. In *2023 IEEE 97th Vehicular Technology Conference (VTC2023-Spring)*, 2023.

[80] P. C. Fishburn. Additive utilities with incomplete product sets: Application to priorities and assignments. *Operations Research*, 15(3):537–542, 1967.

[81] D. J. Fremont, T. Dreossi, S. Ghosh, X. Yue, A. L. Sangiovanni-Vincentelli, and S. A. Seshia. Scenic: A language for scenario specification and scene generation. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*. ACM Press, 2019.

[82] I. M. G. Costantino, M. De Vincenzi. A vehicle firmware security vulnerability: an ivi exploitation. *J Comput Virol Hack Tech*, 20:681,696, 2024.

[83] A. Gazdag, R. Ferenc, and L. Buttyán. Crysys dataset of can traffic logs containing fabrication and masquerade attacks. *Scientific Data*, 10, 2023.

[84] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[85] A. Ghosal and M. Conti. Security issues and challenges in v2x: A survey. *Computer Networks*, 169:107093, 2020.

[86] F. Gonçalves, B. Ribeiro, O. Gama, J. Santos, A. Costa, B. Dias, M. J. Nicolau, J. Macedo, and A. Santos. Synthesizing datasets with security threats for vehicular ad-hoc networks. In *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, pages 1–6, 2020.

[87] J. Gu, M. Bellone, R. Sell, and A. Lind. Object segmentation for autonomous driving using iseauto data. *Electronics*, 11(7), 2022.

[88] D. Guo, Y. Wu, Y. Dai, P. Zhou, X. Lou, and R. Tan. Invisible optical adversarial stripes on traffic sign against autonomous vehicles. In *Proceedings of the 22nd Annual International Conference on Mobile Systems, Applications and Services*, MOBISYS '24, page 534–546, New York, NY, USA, 2024. Association for Computing Machinery.

[89] J. Guo, U. Kurup, and M. Shah. Is it safe to drive? an overview of factors, metrics, and datasets for driveability assessment in autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 21(8):3135–3151, 2020.

[90] J. Guo, L. Li, J. Wang, and K. Li. Cyber-physical system-based path tracking control of autonomous vehicles under cyber-attacks. *IEEE Transactions on Industrial Informatics*, 2023.

[91] Y. Guo, H. Zhang, Z. Li, F. Li, L. Fang, L. Yin, and J. Cao. Decision-Making for Intrusion Response: Which, Where, in What Order, and How Long? In *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, pages 1–6, 2020.

[92] R. S. Hallyburton, Y. Liu, Y. Cao, Z. M. Mao, and M. Pajic. Security analysis of Camera-LiDAR fusion against Black-Box attacks on autonomous vehicles. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1903–1920, Boston, MA, Aug. 2022. USENIX Association.

[93] M. Hamad, A. Finkenzeller, M. Kühr, A. Roberts, O. Maennel, V. Prevelakis, and S. Steinhorst. React: Autonomous intrusion response system for intelligent vehicles. *Computers & Security*, 145:104008, 2024.

[94] M. Hamad, A. Finkenzeller, H. Liu, J. Lauinger, V. Prevelakis, and S. Steinhorst. Seemqtt: Secure end-to-end mqtt-based communication for mobile iot systems using secret sharing and trust delegation. *IEEE Internet of Things Journal*, 10(4):3384–3406, 2023.

[95] M. Hamad, Z. A. Hammadeh, S. Saidi, V. Prevelakis, and R. Ernst. Prediction of abnormal temporal behavior in real-time systems. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, pages 359–367, 2018.

[96] M. Hamad, M. Nolte, and V. Prevelakis. Towards Comprehensive Threat Modeling for Vehicles. In *the 1st Workshop on Security and Dependability of Critical Embedded Real-Time Systems*, 2016.

[97] M. Hamad and V. Prevelakis. SAVTA: A Hybrid Vehicular Threat Model: Overview and Case Study. *Information*, 11(5), 2020.

[98] M. Hamad and S. Steinhorst. Security challenges in autonomous systems design, 2023.

[99] M. Hamad, M. Tsantekidis, and V. Prevelakis. Red-Zone: Towards an Intrusion Response Framework for Intra-Vehicle System. In *5th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS)*, 2019.

[100] M. Hamad, M. Tsantekidis, and V. Prevelakis. Intrusion Response System for Vehicles: Challenges and Vision. In M. Helfert, C. Klein, B. Donnellan, and O. Gusikhin, editors, *Smart Cities, Green Technologies and Intelligent Transport Systems*, pages 321–341, Cham, 2021. Springer International Publishing.

[101] A. Hamdi, M. Muller, and B. Ghanem. SADA: Semantic adversarial diagnostic attacks for autonomous applications. *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence*, pages 10901–10908, 2020.

[102] J. Han, M. Kamber, and J. Pei. 6 - mining frequent patterns, associations, and correlations: Basic concepts and methods. In *Data Mining (Third Edition)*, The Morgan Kaufmann Series in Data Management Systems, pages 243–278. Morgan Kaufmann, Boston, 2012.

[103] J. C. Han and Z. Q. Zhou. *Metamorphic Fuzz Testing of Autonomous Vehicles*, page 380–385. Association for Computing Machinery, New York, NY, USA, 2020.

[104] M. L. Han, B. I. Kwak, and H. K. Kim. Anomaly intrusion detection method for vehicular networks based on survival analysis. *Vehicular Communications*, 14:52–63, 2018.

[105] M. L. Han, B. I. Kwak, and H. K. Kim. Event-triggered interval-based anomaly detection and attack identification methods for an in-vehicle network. *IEEE Transactions on Information Forensics and Security*, 16:2941–2956, 2021.

[106] M. Hanselmann, T. Strauss, K. Dormann, and H. Ulmer. Canet: An unsupervised intrusion detection system for high dimensional can bus data. *IEEE Access*, 8:58194–58205, 2020.

[107] M. R. Heidari Iman, J. Raik, M. Jenihhin, G. Jervan, and T. Ghasempouri. An automated method for mining high-quality assertion sets. *Microprocessors and Microsystems*, 97:104773, 2023.

[108] M. Heigl, L. Doerr, A. Almaini, D. Fiala, and M. Schram. Incident Reaction Based on Intrusion Detections' Alert Analysis. In *2018 International Conference on Applied Electronics (AE)*, pages 1–6, 2018.

[109] O. Henniger, A. Ruddle, H. Seudié, B. Weyl, M. Wolf, and T. Wollinger. Securing vehicular on-board it systems: The evita project. In *VDI/VW Automotive Security Conference*, page 41, 2009.

[110] N. Herold. *Incident Handling Systems with Automated Intrusion Response*. Dissertation, Technische Universität München, 2017.

[111] N. Herold, S.-A. Posselt, O. Hanka, and G. Carle. Anomaly detection for SOME/IP using complex event processing. In *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, pages 1221–1226, 2016.

[112] N. Herold, M. Wachs, S.-A. Posselt, and G. Carle. An Optimal Metric-Aware Response Selection Strategy for Intrusion Response Systems. In F. Cuppens, L. Wang, N. Cuppens-Boulahia, N. Tawbi, and J. Garcia-Alfaro, editors, *Foundations and Practice of Security*, pages 68–84, Cham, 2017. Springer International Publishing.

[113] D. K. Hong, J. Kloosterman, Y. Jin, Y. Cao, Q. A. Chen, S. Mahlke, and Z. M. Mao. AVGuardian: Detecting and Mitigating Publish-Subscribe Overprivilege for Autonomous Vehicle Systems. *Proceedings - 5th IEEE European Symposium on Security and Privacy, Euro S and P 2020*, pages 445–459, 2020.

[114] S. Hu, Y. Zhang, S. Laha, A. Sharma, and H. Foroosh. CCA: Exploring the possibility of contextual camouflage attack on object detection. *Proceedings - International Conference on Pattern Recognition*, pages 7647–7654, 2020.

[115] Z. Hu, J. Shen, S. Guo, X. Zhang, Z. Zhong, Q. A. Chen, and K. Li. Pass: A system-driven evaluation platform for autonomous driving safety and security. *NDSS Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*, 2022.

[116] Y. Huai, Y. Chen, S. Almanee, T. Ngo, X. Liao, Z. Wan, Q. A. Chen, and J. Garcia. Doppelgänger test generation for revealing bugs in autonomous driving software. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 2591–2603, 2023.

[117] K. Hughes, K. McLaughlin, and S. Sezer. Dynamic Countermeasure Knowledge for Intrusion Response Systems. In *2020 31st Irish Signals and Systems Conference (ISSC)*, pages 1–6, 2020.

[118] S. Iannucci, O. D. Barba, V. Cardellini, and I. Banicescu. A performance evaluation of deep reinforcement learning for model-based intrusion response. In *2019 IEEE 4th International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, pages 158–163, 2019.

[119] S. Iannucci, E. Casalicchio, and M. Lucantonio. An Intrusion Response Approach for Elastic Applications Based on Reinforcement Learning. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 01–10, 2021.

[120] S. Iannucci, A. Montemaggio, and B. Williams. Towards self-defense of non-stationary systems. In *2019 International Conference on Computing, Networking and Communications (ICNC)*, pages 250–254, 2019.

[121] C. Iclodean, N. Cordos, and B. O. Varga. Autonomous shuttle bus for public transportation: A review. *Energies*, 13(11), 2020.

[122] I. in Estonia. Estonian auve tech achieves autonomous milestone in japan. `https://estonia.ee/estonian-auve-tech-achieves-autonomous-milestone-in-japan/`, Dec. 2024. Accessed: 2025-04-05.

[123] International Organization for Standardization. *ISO/SAE 21434: 2021: Road Vehicles: Cybersecurity Engineering*. ISO, 2021.

[124] International Telecommunication Union. Guidelines for an intrusion prevention system for connected vehicles - Recommendation ITU-T X.1377, 2022.

[125] S. Iqbal, P. Ball, M. H. Kamarudin, and A. Bradley. Simulating malicious attacks on vanets for connected and autonomous vehicle cybersecurity: A machine learning dataset. In *2022 13th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, pages 332–337, 2022.

[126] M. M. Islam, A. Lautenbach, C. Sandberg, and T. Olovsson. A risk assessment framework for automotive embedded systems. In *Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security*, pages 3–14, 2016.

[127] ISO/TC 22/SC 32 Electrical and electronic components and general system aspects. Iso 26262-1:2018 road vehicles — functional safety. Technical report, International Standards Organization, 2018.

[128] E. F. J. Lu, H. Sibai and D. Forsyth. No need to worry about adversarial examples in object detection in autonomous vehicles. In *in CVPR Workshop of Negative Results in Computer Vision*, 2017.

[129] S. Jeong, S. Lee, H. Lee, and H. K. Kim. X-canids: Signal-aware explainable intrusion detection system for controller area network-based in-vehicle network. *IEEE Transactions on Vehicular Technology*, 73(3):3230–3246, 2024.

[130] S.-Y. Jeong, I.-J. Choi, Y.-J. Kim, Y.-M. Shin, J.-H. Han, G.-H. Jung, and K.-G. Kim. A study on ros vulnerabilities and countermeasure. In *Proceedings of the Companion of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '17, page 147–148, New York, NY, USA, 2017. Association for Computing Machinery.

[131] S. Jha, S. Banerjee, T. Tsai, S. S. Hari, M. B. Sullivan, Z. T. Kalbarczyk, S. W. Keckler, and R. K. Iyer. Ml-based fault injection for autonomous vehicles: A case for bayesian fault injection. In *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 112–124, Los Alamitos, CA, USA, jun 2019. IEEE Computer Society.

[132] S. Jha, S. Cui, S. Banerjee, J. Cyriac, T. Tsai, Z. Kalbarczyk, and R. K. Iyer. ML-Driven Malware that Targets AV Safety. *Proceedings - 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2020*, pages 113–124, 2020.

[133] X. Ji, Y. Cheng, Y. Zhang, K. Wang, C. Yan, W. Xu, and K. Fu. Poltergeist: Acoustic adversarial machine learning against cameras and computer vision. *Proceedings - IEEE Symposium on Security and Privacy*, 2021-May:160–175, 2021.

[134] Y. Jia, Y. Lu, J. Shen, Q. A. Chen, H. Chen, Z. Zhong, and T. Wei. Fooling detection alone is not enough: Adversarial attack against multiple object tracking. In *ICLR*, 2020.

[135] Y. Jia, Y. Lu, J. Shen, Q. A. Chen, H. Chen, Z. Zhong, and T. Wei. Fooling detection alone is not enough: Adversarial attack against multiple object tracking. In *International Conference on Learning Representations*, 2020.

[136] R. Jiao, J. Bai, X. Liu, T. Sato, X. Yuan, Q. A. Chen, and Q. Zhu. Learning representation for anomaly detection of vehicle trajectories. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9699–9706, 2023.

[137] R. Jiao, H. Liang, T. Sato, J. Shen, Q. A. Chen, and Q. Zhu. End-to-end uncertainty-based mitigation of adversarial attacks to automated lane centering. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, 2021.

[138] P. Jing, Q. Tang, Y. Du, L. Xue, X. Luo, T. Wang, S. Nie, and S. Wu. Too good to be safe: Tricking lane detection in autonomous driving with crafted perturbations. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 3237–3254. USENIX Association, Aug. 2021.

[139] H. J. Jo and W. Choi. A survey of attacks on controller area networks and corresponding countermeasures. *IEEE Transactions on Intelligent Transportation Systems*, 23(7):6123–6141, 2022.

[140] 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*, 7(4):815–837, 2022.

[141] C. Jurewicz, A. Sobhani, J. Woolley, J. Dutschke, and B. Corben. Exploration of Vehicle Impact Speed – Injury Severity Relationships for Application in Safer Road Design. *Transportation Research Procedia*, 14:4247–4256, 2016.

[142] A. Karahasanovic, P. Kleberger, and M. Almgren. Adapting Threat Modeling Methods for the Automotive Industry. In *ej tryckt*, 2017.

[143] S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa, A. Monrroy, T. Ando, Y. Fujii, and T. Azumi. Autoware on board: Enabling autonomous vehicles with embedded systems. In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, pages 287–296, 2018.

[144] S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa, A. Monrroy, T. Ando, Y. Fujii, and T. Azumi. Autoware on board: Enabling autonomous vehicles with embedded systems. In *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, pages 287–296. IEEE, 2018.

[145] S. Katsikeas, P. Johnsson, S. Hacks, and R. Lagerström. Vehiclelang: A probabilistic modeling and simulation language for modern vehicle it infrastructures. *Computers Security*, 117:102705, 2022.

[146] P. Kaur, S. Taghavi, Z. Tian, and W. Shi. A survey on simulators for testing self-driving cars. In *2021 Fourth International Conference on Connected and Autonomous Driving (MetroCAD)*, pages 62–70, 2021.

[147] D. Kerr. Protesters stop waymo and cruise self-driving cars with only a traffic cone. *NPR*, Aug. 2023. Accessed: 2025-04-05.

[148] S. K. Khan, N. Shiwakoti, P. Stasinopoulos, and Y. Chen. Cyber-attacks in the next-generation cars, mitigation techniques, anticipated readiness and future directions. *Accident Analysis Prevention*, 148:105837, 2020.

[149] H. A. Kholidy, A. Erradi, S. Abdelwahed, and F. Baiardi. A risk mitigation approach for autonomous cloud intrusion response system. *Computing*, 98(11):1111–1135, Nov 2016.

[150] H. Kim, R. Bandyopadhyay, M. Ozmen, Z. Celik, A. Bianchi, Y. Kim, and D. Xu. A systematic study of physical sensor attack hardness. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 146–146, Los Alamitos, CA, USA, may 2024. IEEE Computer Society.

[151] H. Kim, M. O. Ozmen, A. Bianchi, Z. B. Celik, and D. Xu. Pgfuzz: Policy-guided fuzzing for robotic vehicles. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2021.

[152] K. Kim, J. S. Kim, S. Jeong, J.-H. Park, and H. K. Kim. Cybersecurity for autonomous vehicles: Review of attacks and defense. *Computers & Security*, 103:102150, 2021.

[153] S. Kim and T. Kim. Robofuzz: fuzzing robotic systems over robot operating system (ros) for finding correctness bugs. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2022, page 447–458, New York, NY, USA, 2022. Association for Computing Machinery.

[154] S. Kim, M. Liu, J. J. Rhee, Y. Jeon, Y. Kwon, and C. H. Kim. Drivefuzz: Discovering autonomous driving bugs through driving quality-guided fuzzing. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CSS '22)*. ACM Press, 2022.

[155] T. Kim, C. H. Kim, J. Rhee, F. Fei, Z. Tu, G. Walkup, X. Zhang, X. Deng, and D. Xu. RVFuzzer: Finding input validation bugs in robotic vehicles through Control-Guided testing. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 425–442, Santa Clara, CA, Aug. 2019. USENIX Association.

[156] V. Klee and G. J. Minty. How good is the simplex algorithm? In *Inequalities III (Proc. Third Sympos., Univ. California, Los Angeles, Calif., 1969; dedicated to the memory of Theodore S. Motzkin)*, page 159–175, New York, 1972. Academic Press.

[157] M. Klischat and M. Althoff. Generating critical test scenarios for automated vehicles with evolutionary algorithms. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 2352–2358, 2019.

[158] M. Kneib and C. Huth. Scission: Signal characteristic-based sender identification and intrusion detection in automotive networks. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 787–800, 2018.

[159] S. Köhler, G. Lovisotto, S. Birnbach, R. Baker, and I. Martinovic. They See Me Rollin : Inherent Vulnerability of the Rolling Shutter in CMOS Image Sensors. *ACM International Conference Proceeding Series*, pages 399–413, 2021.

[160] A. Konak, D. W. Coit, and A. E. Smith. Multi-objective optimization using genetic algorithms: A tutorial. *Reliability engineering & system safety*, 91(9):992–1007, 2006.

[161] Z. Kong, J. Guo, A. Li, and C. Liu. PhysGAN: Generating Physical-World-Resilient Adversarial Examples for Autonomous Driving. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 14242–14251, 2020.

[162] M. Kühr, M. Hamad, P. MohajerAnsari, M. D. Pesé, and S. Steinhorst. Sok: Security of the image processing pipeline in autonomous vehicles, 2024.

[163] L. Huang, C. Gao, Y. Zhou, C. Xie, A. L. Yuille, C. Zou, and N. Liu,. Universal physical camouflage attacks on object detectors. In *in CVPR*, 2020.

[164] H. S. Lallie, K. Debattista, and J. Bal. A review of attack graph and attack tree visual syntax in cyber security. *Computer Science Review*, 2020.

[165] B. Lampe and W. Meng. can-train-and-test: A new can intrusion detection dataset. In *2023 IEEE 98th Vehicular Technology Conference (VTC2023-Fall)*, pages 1–7, 2023.

[166] B. Lampe and W. Meng. can-train-and-test: A curated can dataset for automotive intrusion detection. *Computers & Security*, 140:103777, 2024.

[167] J. Lauinger, A. Finkenzeller, H. Lautebach, M. Hamad, and S. Steinhorst. Attack data generation framework for autonomous vehicle sensors. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 128–131, 2022.

[168] A. Lautenbach, M. Almgren, and T. Olovsson. Proposing heavens 2.0–an automotive risk assessment model. In *Proceedings of the 5th ACM Computer Science in Cars Symposium*, pages 1–12, 2021.

[169] S. Lee, W. Choi, I. Kim, G. Lee, and D. H. Lee. A comprehensive analysis of datasets for automotive intrusion detection systems. *Computers, Materials and Continua*, 76(3):3413–3442, 2023.

[170] G. Li, Y. Li, S. Jha, T. Tsai, M. Sullivan, S. K. S. Hari, Z. Kalbarczyk, and R. Iyer. Av-fuzzer: Finding safety violations in autonomous driving systems. In *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, pages 25–36, 2020.

[171] Y. Li, C. Wen, F. Juefei-Xu, and C. Feng. Fooling LiDAR Perception via Adversarial Trajectory Perturbation. pages 7878–7887, 2022.

[172] T. Limbasiya, K. Z. Teng, S. Chattopadhyay, and J. Zhou. A systematic survey of attack detection and prevention in connected and autonomous vehicles. *Vehicular Communications*, 37:100515, 2022.

[173] X. Liu, R. Jiao, B. Zheng, D. Liang, and Q. Zhu. Safety-driven interactive planning for neural network-based lane changing. In *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, ASPDAC '23. Association for Computing Machinery, 2023.

[174] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang. Trojaning attack on neural networks. In *NDSS*. The Internet Society, 2018.

[175] S.-F. Lokman, A. T. Othman, and M.-H. Abu-Bakar. Intrusion detection system for automotive Controller Area Network (CAN) bus system: a review. *EURASIP Journal on Wireless Communications and Networking*, 2019(1):184, Jul 2019.

[176] A. Lopes and A. Hutchison. Experimenting with Machine Learning in Automated Intrusion Response. In I. Kotenko, C. Badica, V. Desnitsky, D. El Baz, and M. Ivanovic, editors, *Intelligent Distributed Computing XIII*, pages 505–514, Cham, 2020. Springer International Publishing.

[177] G. Lou, Y. Deng, X. Zheng, M. Zhang, and T. Zhang. Testing of autonomous driving systems: where are we and where should we go? In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2022, page 31–43, New York, NY, USA, 2022. Association for Computing Machinery.

[178] G. Lovisotto, H. Turner, I. Sluganovic, M. Strohmeier, and I. Martinovic. SLAP: Improving physical adversarial examples with short-lived adversarial perturbations. *Proceedings of the 30th USENIX Security Symposium*, pages 1865–1882, 2021.

[179] F. Luo, Y. Jiang, Z. Zhang, Y. Ren, and S. Hou. Threat Analysis and Risk Assessment for Connected Vehicles: A Survey. *Security and Communication Networks*, 2021, Sep 2021.

[180] M. Luo, A. C. Myers, and G. E. Suh. Stealthy tracking of autonomous vehicles with cache side channels. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 859–876. USENIX Association, Aug. 2020.

[181] C. Ma, N. Wang, Q. A. Chen, and C. Shen. SlowTrack: Increasing the Latency of Camera-Based Perception in Autonomous Driving Using Adversarial Examples. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 4062–4070, 2024.

[182] C. Ma, N. Wang, Q. A. Chen, and C. Shen. Slowtrack: Increasing the latency of camera-based perception in autonomous driving using adversarial examples. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(5):4062–4070, Mar. 2024.

[183] Y. Ma, J. Sharp, R. Wang, E. Fernandes, and X. Zhu. Sequential attacks on kalman filter-based forward collision warning systems. In *AAAI Conference on Artificial Intelligence*, 2020.

[184] K. T. Y. Mahima, M. Ayoob, and G. Poravi. Adversarial Attacks and Defense Technologies on Autonomous Vehicles: A Review. *Applied Computer Systems*, 26(2):96–106, 2021.

[185] E. Malayjerdi, R. Sell, M. Malayjerdi, A. Udal, and M. Bellone. Practical path planning techniques in overtaking for autonomous shuttles. *Journal of Field Robotics*, 39(4):410–425, 2022.

[186] E. Malayjerdi, R. Sell, M. Malayjerdi, A. Udal, and M. Bellone. Practical path planning techniques in overtaking for autonomous shuttles. *Journal of Field Robotics*, 39(4):410–425, 2022.

[187] M. Malayjerdi, V. Kuts, R. Sell, T. Otto, and B. C. Baykara. Virtual simulations environment development for autonomous vehicles interaction. In *ASME International Mechanical Engineering Congress and Exposition*. American Society of Mechanical Engineers, 2020.

[188] M. Malayjerdi, A. Roberts, O. Maennel, and E. Malayjerdi. Combined Safety and Cybersecurity Testing Methodology for Autonomous Driving Algorithms, 2022.

[189] M. Malayjerdi, A. Roberts, O. M. Maennel, and E. Malayjerdi. Combined safety and cybersecurity testing methodology for autonomous driving algorithms. *Proceedings of the 6th ACM Computer Science in Cars Symposium*, pages 1–10, 2022.

[190] M. Malayjerdi, A. Roberts, O. m. Maennel, and E. Malayjerdi. Combined safety and cybersecurity testing methodology for autonomous driving algorithms. In *Proceedings of the 6th ACM Computer Science in Cars Symposium*, CSCS '22, New York, NY, USA, 2022. Association for Computing Machinery.

[191] Y. Man, M. Li, and R. Gerdes. GhostImage: Remote perception attacks against camera-based image classification systems. *RAID 2020 Proceedings - 23rd International Symposium on Research in Attacks, Intrusions and Defenses*, pages 317–332, 2020.

[192] S. F. Marksteiner, C. Schmittner, K. Christl, D. Nickovic, M. Sjödin, and M. Sirjani. From tara to test: Automated automotive cybersecurity test generation out of threat modeling. In *Proceedings of the 7th ACM Computer Science in Cars Symposium*, CSCS '23, New York, NY, USA, 2023. Association for Computing Machinery.

[193] G. Matthews and B. Feinstein. The Intrusion Detection Exchange Protocol (IDXP). RFC 4767, Mar. 2007.

[194] Mcity. Mcity, 2025. Accessed: 2024-03-17.

[195] S. Meryem and T. Mazri. Security study and challenges of connected autonomous vehicles. In *Proceedings of the 4th International Conference on Smart City Applications*, SCA '19, New York, NY, USA, 2019. Association for Computing Machinery.

[196] C. Miller and C. Valasek. Remote exploitation of an unaltered passenger vehicle. *Black Hat USA*, 2015.

[197] C. Miller and C. Valasek. Remote Exploitation of an Unaltered Passenger Vehicle. `https://illmatics.com/Remote\%20Car\%20Hacking.pdf`, 2015. Accessed: 12.04.2022.

[198] R. Mit, Y. Zangvil, and D. Katalan. Analyzing tesla's level 2 autonomous driving system under different gnss spoofing scenarios and implementing connected services for authentication and reliability of gnss data. In *Proceedings of the 33rd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2020)*, pages 621–646, September 2020.

[199] S. Mitchell, M. O'Sullivan, and I. Dunning. PuLP: A Linear Programming Toolkit for Python. *Department of Engineering Science, The University of Auckland, Auckland, New Zealand*, 2011.

[200] Mohammed Lamine Bouchouia, Jean-Philippe Monteuuis, Houda Labiod, Ons Jelassi, Wafa Ben Jaballah , Jonathan Petit. A simulator for cooperative and automated driving security. In *Fourth International Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*, 2022.

[201] T. Mokhamed, F. M. Dakalbab, S. Abbas, and M. A. Talib. Security in robot operating systems (ros): analytical review study. In *The 3rd International Conference on Distributed Sensing and Intelligent Systems (ICDSIS 2022)*, volume 2022, pages 79–94, 2022.

[202] L. J. Moukahal, M. Zulkernine, and M. Soukup. Boosting grey-box fuzzing for connected autonomous vehicle systems. In *2021 IEEE 21st International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pages 516–527, 2021.

[203] J. Mulach. How a t-shirt stopped this autonomous car in its tracks. `https://www.carexpert.com.au/car-news/how-a-t-shirt-stopped-this-autonomous-car-in-its-tracks`, May 2024. Accessed: 2025-04-05.

[204] M. Müller, V. Casser, J. Lahoud, N. Smith, and B. Ghanem. Sim4cv: A photo-realistic simulator for computer vision applications. *Int. J. Comput. Vision*, 126(9):902–919, sep 2018.

[205] F. Munir, S. Azam, M. I. Hussain, A. M. Sheri, and M. Jeon. Autonomous vehicle: The architecture aspect of self driving car. In *Proceedings of the 2018 International Conference on Sensors, Signal and Image Processing*, SSIP '18, page 1–5, New York, NY, USA, 2018. Association for Computing Machinery.

[206] K. K. Nakka and M. Salzmann. Indirect Local Attacks for Context-Aware Semantic Segmentation Networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12350 LNCS:611–628, 2020.

[207] P. Narksri, H. Darweesh, E. Takeuchi, Y. Ninomiya, and K. Takeda. Occlusion-aware motion planning with visibility maximization via active lateral position adjustment. *IEEE Access*, 10:57759–57782, 2022.

[208] B. Nassi, R. Bitton, R. Masuoka, A. Shabtai, and Y. Elovici. Sok: Security and privacy in the age of commercial drones. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1434–1451, 2021.

[209] 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. *Proceedings of the ACM Conference on Computer and Communications Security*, pages 293–308, 2020.

[210] 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*, CCS '20, page 293–308, New York, NY, USA, 2020. Association for Computing Machinery.

[211] P. Nespoli, D. Papamartzivanos, F. Gómez Mármol, and G. Kambourakis. Optimal countermeasures selection against cyber attacks: A comprehensive survey on reaction frameworks. *IEEE Communications Surveys & Tutorials*, 20(2):1361–1396, 2018.

[212] F. Nesti, G. Rossolini, S. Nair, A. Biondi, and G. Buttazzo. Evaluating the Robustness of Semantic Segmentation for Autonomous Driving against Real-World Adversarial Patch Attacks. *Proceedings - 2022 IEEE/CVF Winter Conference on Applications of Computer Vision, WACV 2022*, pages 2826–2835, 2022.

[213] E. C. P. Neto, H. Taslimasa, S. Dadkhah, S. Iqbal, P. Xiong, T. Rahman, and A. A. Ghorbani. Ciciov2024: Advancing realistic ids approaches against dos and spoofing attack in iov can bus. *Internet of Things*, 26:101209, 2024.

[214] N. News. Lyft to roll out robotaxis in atlanta. `https://www.nbcnews.com/tech/tech-news/lyft-roll-robotaxis-atlanta-rcna197479`, Mar. 2025. Accessed: 2025-04-05.

[215] M. O'Kelly, A. Sinha, H. Namkoong, J. Duchi, and R. Tedrake. Scalable end-to-end autonomous vehicle testing via rare-event simulation. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 9849–9860, Red Hook, NY, USA, 2018. Curran Associates Inc.

[216] C. Olt. Establishing security operation centers for connected cars. *ATZelectronics worldwide*, 14(5):40–43, 2019.

[217] S. Ossenbühl, J. Steinberger, and H. Baier. Towards Automated Incident Handling: How to Select an Appropriate Response against a Network-Based Attack? In *2015 Ninth International Conference on IT Security Incident Management & IT Forensics*, pages 51–67, 2015.

[218] A. Palanca, E. Evenchick, F. Maggi, and S. Zanero. A stealth, selective, link-layer denial-of-service attack against automotive networks. In *Detection of Intrusions and Malware, and Vulnerability Assessment: 14th International Conference, DIMVA 2017, Bonn, Germany, July 6-7, 2017, Proceedings 14*, pages 185–206. Springer, 2017.

[219] M. Papadaki, S. Furnell, B. Lines, and P. Reynolds. Operational Characteristics of an Automated Intrusion Response System. In A. Lioy and D. Mazzocchi, editors, *Communications and Multimedia Security. Advanced Techniques for Network and Data Protection*, pages 65–75, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[220] S. Parkinson, P. Ward, K. Wilson, and J. Miller. Cyber threats facing autonomous and connected vehicles: Future challenges. *IEEE Transactions on Intelligent Transportation Systems*, 18(11):2898–2915, 2017.

[221] K. Paul. The rebel group stopping self-driving cars in san francisco – one cone at a time. *The Guardian*, July 2023. Accessed: 2025-04-05.

[222] M. D. Pesé, T. Stacer, C. A. Campos, E. Newberry, D. Chen, and K. G. Shin. Librecan: Automated can message translator. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 2283–2300, New York, NY, USA, 2019. Association for Computing Machinery.

[223] J. Petit and S. E. Shladover. Potential cyberattacks on automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):546–556, 2015.

[224] Petter Solnør,Øystein Volden,Kristoffer Gryte,Slobodan Petrovic,Thor I. Fossen. Hijacking of unmanned surface vehicles: A demonstration of attacks and countermeasures in the field. *Journal of Field Robotics*, pages 1–19, 2022.

[225] O. Pöllny, F. Kargl, and A. Held. Steering your car with electromagnetic fields. In *Proceedings of the 6th ACM Computer Science in Cars Symposium*, CSCS '22, New York, NY, USA, 2022. Association for Computing Machinery.

[226] P. Project. Platform for innovative use of vehicle open telematics. 2024.

[227] Python Software Foundation. resource — Resource usage information. `https://docs.python.org/3/library/resource.html`, 2022. Accessed: 20.07.2022.

[228] R. Quinonez, J. Giraldo, L. Salazar, E. Bauman, A. Cardenas, and Z. Lin. Savior: securing autonomous vehicles with robust physical invariants. In *Proceedings of the 29th USENIX Conference on Security Symposium*, SEC'20, USA, 2020. USENIX Association.

[229] S. Rajapaksha, G. Madzudzo, H. Kalutarage, A. Petrovski, and M. O. Al-Kadri. Canmirgu: A comprehensive can bus attack dataset from moving vehicles for intrusion detection system evaluation. In *Symposium on Vehicles Security and Privacy. Internet Society*, 2024.

[230] K. Ren, Q. Wang, C. Wang, Z. Qin, and X. Lin. The security of autonomous driving: Threats, defenses, and future directions. *Proceedings of the IEEE*, 108(2):357–372, 2020.

[231] D. C. Richards. Relationship between speed and risk of fatal injury: pedestrians and car occupants. In *Road Safety Web Publication*, volume No. 16, London, 2010. Department for Transport.

[232] S. Rivera, A. K. Iannillo, and R. State. Discofuzzer: Discontinuity-based vulnerability detector for robotic systems. July 2020.

[233] A. Roberts, J. Cheng, O. Maennel, M. Hamad, and S. Steinhorst. Adseclang: A domain-specific language for cybersecurity testing of autonomous vehicles. In *2025 IEEE 101st Vehicular Technology Conference (VTC2025-Spring)*, pages 1–6, 2025.

[234] A. Roberts, M. R. H. Iman, M. Bellone, T. Ghasempouri, J. Raik, O. Maennel, M. Hamad, and S. Steinhorst. Adassure: Debugging methodology for autonomous driving control algorithms. In *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6, 2024.

[235] A. Roberts, O. Maennel, and N. Snetkov. Cybersecurity test range for autonomous vehicle shuttles. *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*, pages 239–248, 2021.

[236] A. Roberts, M. Malayjerdi, M. Bellone, O. Maennel, and E. Malayjerdi. Analysing adversarial threats to rule-based local-planning algorithms for autonomous driving. *Inaugural Symposium on Vehicle Security and Privacy (VehicleSec 2023) with NDSS*, pages 1–8, 2023.

[237] A. Roberts, M. Malayjerdi, M. Bellone, R. Sell, O. Maennel, M. Hamad, and S. Steinhorst. Adsecdata platform: An open-source data platform for autonomous driving cybersecurity. In *2025 IEEE 101st Vehicular Technology Conference (VTC2025-Spring)*, pages 1–7, 2025.

[238] A. Roberts, M. Malayjerdi, M. Bellone, R. Sell, O. Maennel, M. Hamad, and S. Steinhorst. Analysis of autonomous driving software to low-level sensor cyber attacks. In *2025 IEEE/ACM 20th Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 1–11, 2025.

[239] A. Roberts, S. Marksteiner, M. Soyturk, B. Yaman, and Y. Yang. A global survey of standardization and industry practices of automotive cybersecurity validation and verification testing processes and tools. *SAE International Journal of Connected and Automated Vehicles*, 7, 11 2023.

[240] A. Roberts, L. Teply, M. Bellone, M. Pese, O. Maennel, M. Hamad, and S. Steinhorst. Fuzzsense: Towards a modular fuzzing framework for autonomous driving software. In *arXiv*, 2025.

[241] G. Rong, B. H. Shin, H. Tabatabaee, Q. Lu, S. Lemke, M. Možeiko, E. Boise, G. Uhm, M. Gerow, S. Mehta, E. Agafonov, T. H. Kim, E. Sterner, K. Ushiroda, M. Reyes, D. Zelenkovsky, and S. Kim. Lgsvl simulator: A high fidelity simulator for autonomous driving. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, page 1–6. IEEE Press, 2020.

[242] G. Rong, B. H. Shin, H. Tabatabaee, Q. Lu, S. Lemke, M. Možeiko, E. Boise, G. Uhm, M. Gerow, S. Mehta, E. Agafonov, T. H. Kim, E. Sterner, K. Ushiroda, M. Reyes, D. Zelenkovsky, and S. Kim. Lgsvl simulator: A high fidelity simulator for autonomous driving. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, 2020.

[243] J. R. Rose, M. Swann, K. P. Grammatikakis, I. Koufos, G. Bendiab, S. Shiaeles, and N. Kolokotronis. Ideres: Intrusion detection and response system using machine learning and attack graphs. *Journal of Systems Architecture*, 131:102722, 2022.

[244] L. A. Rosero, I. P. Gomes, J. A. R. da Silva, C. A. Przewodowski, D. F. Wolf, and F. S. Osório. Integrating modular pipelines with end-to-end learning: A hybrid approach for robust and reliable autonomous driving systems. *Sensors*, 24(7), 2024.

[245] A. Rugo, C. A. Ardagna, and N. E. Ioini. A security review in the uavnet era: Threats, countermeasures, and gap analysis. *ACM Comput. Surv.*, 55(1), jan 2022.

[246] SAE International. J3016 taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. Technical report, Society of Automotive Engineers, 2021.

[247] I. F. Salgado, N. Quijano, D. J. Fremont, and A. A. Cardenas. Fuzzing malicious driving behavior to find vulnerabilities in collision avoidance systems. In *2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 368–375, 2022.

[248] T. Sato, S. H. Bhupathiraju, M. Clifford, T. Sugawara, Q. A. Chen, and S. Rampazzi. Wip: Infrared laser reflection attack against traffic sign recognition systems. *ISOC Symposium on Vehicle Security and Privacy (VehicleSec)*.

[249] T. Sato, J. Shen, N. Wang, Y. Jia, X. Lin, and Q. A. Chen. Dirty road can attack: Security of deep learning based automated lane centering under Physical-World attack. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 3309–3326. USENIX Association, Aug. 2021.

[250] T. Sato, J. Shen, N. Wang, Y. J. Jia, X. Lin, and Q. A. Chen. Demo: Security of Deep Learning based Automated Lane Centering under Physical-World Attack. *Proceedings - 2021 IEEE Symposium on Security and Privacy Workshops, SPW 2021*, page 244, 2021.

[251] 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 *ISOC Network and Distributed System Security Symposium (NDSS)*, 2025.

[252] T. Sato, J. Yue, N. Chen, N. Wang, and Q. A. Chen. Intriguing Properties of Diffusion Models: An Empirical Study of the Natural Attack Capability in Text-to-Image Generative Models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[253] A. Schrijver. The simplex method. In *Theory of Linear and Integer Programming*, pages 129–150, New York, 1998. John Wiley & Sons.

[254] R. Sell, M. Leier, A. Rassõlkin, and J.-P. Ernits. Self-driving car iseauto for research and education. In *2018 19th International Conference on Research and Education in Mechatronics (REM)*, pages 111–116, 2018.

[255] R. Sell, E. Malayjerdi, M. Malayjerdi, and B. C. Baykara. Safety toolkit for automated vehicle shuttle -practical implementation of digital twin. In *2022 International Conference on Connected Vehicle and Expo (ICCVE)*, pages 1–6, 2022.

[256] R. Sell, M. Malayjerdi, H. Pikner, R. Razdan, E. Malayjerdi, and M. Bellone. Open-source level 4 autonomous shuttle for last - mile mobility. In *2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 01–06, 2024.

[257] V. Sembera. Iso/sae 21434: Setting the standard for connected cars' cybersecurity. *Trend Micro Research*, *White Paper*, 2020.

[258] E. Seo, H. M. Song, and H. K. Kim. Gids: Gan based intrusion detection system for in-vehicle network. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, pages 1–6, Aug 2018.

[259] S. Shah, D. Dey, C. Lovett, and A. Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In M. Hutter and R. Siegwart, editors, *Field and Service Robotics*, pages 621–635, Cham, 2018. Springer International Publishing.

[260] M. Shahin, M. R. Heidari Iman, M. Kaushik, R. Sharma, T. Ghasempouri, and D. Draheim. Exploring factors in a crossroad dataset using cluster-based association rule mining. *Procedia Computer Science*, 2022.

[261] A. Shameli-Sendi, N. Ezzati-Jivan, M. Jabbarifar, and M. Dagenais. Intrusion Response Systems: Survey and Taxonomy. *International Journal Computer Science Network Security (IJCSNS)*, 12, 2012.

[262] J. Shen, Y. Luo, Z. Wan, and Q. A. Chen. Lateral-direction localization attack in high-level autonomous driving: Domain-specific defense opportunity via lane detection, 2023.

[263] J. Shen, N. Wang, Z. Wan, Y. Luo, T. Sato, Z. Hu, X. Zhang, S. Guo, Z. Zhong, K. Li, Z. Zhao, C. Qiao, and Q. A. Chen. Sok: On the semantic ai security in autonomous driving, 2024.

[264] J. Shen, J. Y. Won, Z. Chen, and Q. A. Chen. Drift with devil: Security of multi-sensor fusion based localization in high-level autonomous driving under GPS spoofing. *Proceedings of the 29th USENIX Security Symposium*, pages 931–948, 2020.

[265] L. Sherry, J. Shortle, G. Donohue, B. Berlin, and J. West. Autonomous systems design, testing, and deployment: Lessons learned from the deployment of an autonomous shuttle bus. In *2020 Integrated Communications Navigation and Surveillance Conference (ICNS)*, pages 5D1–1–5D1–14, 2020.

[266] Y. Son, H. Shin, D. Kim, Y. Park, J. Noh, K. Choi, J. Choi, and Y. Kim. Rocking drones with intentional sound noise on gyroscopic sensors. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 881–896, Washington, D.C., Aug. 2015. USENIX Association.

[267] D. Song, K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, F. Tramèr, A. Prakash, and T. Kohno. Physical adversarial examples for object detectors. In *12th USENIX Workshop on Offensive Technologies (WOOT 18)*, Baltimore, MD, Aug. 2018. USENIX Association.

[268] H. M. Song, J. Woo, and H. K. Kim. In-vehicle network intrusion detection using deep convolutional neural network. *Vehicular Communications*, 21:100198, 2020.

[269] S. Souissi, Serhrouchni, L. Sliman, and B. Charroux. Security Incident Response: Towards a Novel Decision-Making System. In A. M. Madureira, A. Abraham, D. Gamboa, and P. Novais, editors, *Intelligent Systems Design and Applications*. Springer International Publishing, 2017.

[270] N. Stakhanova, S. Basu, and J. Wong. A taxonomy of intrusion response system. *International Journal of Information and Computer Security*, 1:169–184, 2007.

[271] N. Stakhanova, C. Strasburg, S. Basu, and J. S. Wong. Towards cost-sensitive assessment of intrusion response selection. *Journal of computer security*, 20(2-3):169–198, 2012.

[272] C. Strasburg, N. Stakhanova, S. Basu, and J. S. Wong. A framework for cost sensitive assessment of intrusion response selection. In *2009 33rd Annual IEEE international computer software and applications conference*, volume 1, pages 355–360. IEEE, 2009.

[273] J. Sun, Y. Cao, Q. A. Chen, and Z. Morley Mao. Towards robust LiDAR-based perception in autonomous driving: General black-box adversarial sensor attack and countermeasures. *Proceedings of the 29th USENIX Security Symposium*, pages 877–894, 2020.

[274] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[275] Z. Sun, S. Balakrishnan, L. Su, A. Bhuyan, P. Wang, and C. Qiao. Who Is in Control? Practical Physical Layer Attack and Defense for mmWave-Based Sensing in Autonomous Vehicles. *IEEE Transactions on Information Forensics and Security*, 16:3199–3214, 2021.

[276] K. Tam and K. Jones. Cyber-risk assessment for autonomous ships. In *2018 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, pages 1–8, 2018.

[277] K. Tang, J. S. Shen, and Q. A. Chen. Fooling perception via location: A case of region-of-interest attacks on traffic light detection in autonomous driving. *NDSS Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*.

[278] M. M. Tani, Jacopo et al. "Duckietown: An Innovative Way to Teach Autonomy." Alimisis D. and e. Menegatti E. Duckietown: An innovative way to teach autonomy. In *Educational Robotics in the Makers Era. Edurobotics 2016. Advances in Intelligent Systems and Computing 560*, pages 104–121. Springer International Publishing AG, 2017.

[279] The Waymo Team. Waymo one is now open to everyone in san francisco. `https://waymo.com/blog/2024/06/waymo-one-is-now-open-to-everyone-in-san-francisco`, June 2024. Accessed: 2025-04-05.

[280] V. L. Thing and J. Wu. Autonomous vehicle security: A taxonomy of attacks and defences. In *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 164–170, 2016.

[281] Y. Tian, K. Pei, S. Jana, and B. Ray. Deeptest: automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th International Conference on Software Engineering*, ICSE '18, page 303–314, New York, NY, USA, 2018. Association for Computing Machinery.

[282] ToyotaInfoTech. Resistant automotive miniature network. *Toyota Info Tech*, 2024.

[283] T. Trippel, O. Weisse, W. Xu, P. Honeyman, and K. Fu. Walnut: Waging doubt on the integrity of mems accelerometers with acoustic injection attacks. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 3–18, 2017.

[284] T. Tsai, K. Yang, T. Y. Ho, and Y. Jin. Robust adversarial objects against deep learning models. *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence*, pages 954–962, 2020.

[285] K.-L. TSUI. An overview of taguchi method and newly developed statistical methods for robust design. *IIE Transactions*, 24(5):44–57, 1992.

[286] J. Tu, H. Li, X. Yan, M. Ren, Y. Chen, M. Liang, E. Bitar, E. Yumer, and R. Urtasun. Exploring Adversarial Robustness of Multi-Sensor Perception Systems in Self Driving. (CoRL):1–12, 2021.

[287] J. Tu, H. Li, X. Yan, M. Ren, Y. Chen, M. Liang, E. Bitar, E. Yumer, and R. Urtasun. Exploring adversarial robustness of multi-sensor perception systems in self driving. In *5th Annual Conference on Robot Learning*, 2021.

[288] J. Tu, H. Li, X. Yan, M. Ren, Y. Chen, M. Liang, E. Bitar, E. Yumer, and R. Urtasun. Exploring adversarial robustness of multi-sensor perception systems in self driving. In A. Faust, D. Hsu, and G. Neumann, editors, *Proceedings of the 5th Conference on*

Robot Learning, volume 164 of *Proceedings of Machine Learning Research*, pages 1013–1024. PMLR, 08–11 Nov 2022.

[289] Y. Tu, Z. Lin, I. Lee, and X. Hei. Injected and delivered: Fabricating implicit control over actuation systems by spoofing inertial sensors. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1545–1562, Baltimore, MD, Aug. 2018. USENIX Association.

[290] Y. Tu, V. S. Tida, Z. Pan, and X. Hei. Transduction shield: A low-complexity method to detect and correct the effects of emi injection attacks on sensors. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, ASIA CCS '21, page 901–915, New York, NY, USA, 2021. Association for Computing Machinery.

[291] Turla. Mazda getinfo attack. 2017.

[292] S. Ullah, M. A. Khan, J. Ahmad, S. S. Jamal, Z. e Huma, M. T. Hassan, N. Pitropakis, Arshad, and W. J. Buchanan. HDL-IDS: A Hybrid Deep Learning Architecture for Intrusion Detection in the Internet of Vehicles. *Sensors*, 22(4), 2022.

[293] S. Ullah, S. Shelly, A. Hassanzadeh, A. Nayak, and K. Hasan. On the Effectiveness of Intrusion Response Systems against Persistent Threats. In *2020 International Conference on Computing, Networking and Communications (ICNC)*, pages 415–421, 2020.

[294] N. United States National Transportation Safety Board. Investigation of lion air flight 610 and ethiopian airlines flight 302. *Safety Recommendation Report NTSB ASR1901. PDF document*, 2019.

[295] Upstream. Upstream's 2022 global automotive cybersecurity report, 2022.

[296] A. Vahidi, T. Rosenstatter, and N. I. Mowla. Systematic evaluation of automotive intrusion detection datasets. In *Proceedings of the 6th ACM Computer Science in Cars Symposium*, CSCS '22, New York, NY, USA, 2022. Association for Computing Machinery.

[297] N. Vinzenz and D. Oka. Integrating fuzz testing into the cyber- security validation strategy. *SAE Technical Paper*, 01(0139), 2021.

[298] Z. Wan, J. Shen, J. Chuang, X. Xia, J. Garcia, J. Ma, and Q. A. Chen. Too Afraid to Drive: Systematic Discovery of Semantic DoS Vulnerability in Autonomous Driving Planning under Physical-World Attacks. In *Network and Distributed System Security (NDSS) Symposium, 2022*, April 2022.

[299] B. Wang, Y. Sun, M. Sun, and X. Xu. Game-Theoretic Actor–Critic-Based Intrusion Response Scheme (GTAC-IRS) for Wireless SDN-Based IoT Networks. *IEEE Internet of Things Journal*, 8(3):1830–1845, 2021.

[300] C. Wang, X. Wang, H. Hu, and et al. On the application of cameras used in autonomous vehicles. *Arch Computational Methods in Engineering*, 29(5):4319–4339, 2022.

[301] D. Wang, C. Li, S. Wen, Q. L. Han, S. Nepal, X. Zhang, and Y. Xiang. Daedalus: Breaking Nonmaximum Suppression in Object Detection via Adversarial Examples. *IEEE Transactions on Cybernetics*, pages 1–14, 2021.

[302] J. Wang, A. Liu, Z. Yin, S. Liu, S. Tang, and X. Liu. Dual Attention Suppression Attack: Generate Adversarial Camouflage in Physical World. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 8561–8570, 2021.

[303] J. Wang, A. Pun, J. Tu, S. Manivasagam, A. Sadat, S. Casas, M. Ren, and R. Urtasun. Advsim: Generating safety-critical scenarios for self-driving vehicles. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[304] P. Wang, X. Huang, X. Cheng, D. Zhou, Q. Geng, and R. Yang. The apolloscape open dataset for autonomous driving and its application. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[305] W. Wang, Y. Yao, X. Liu, X. Li, P. Hao, and T. Zhu. I Can See the Light: Attacks on Autonomous Vehicles Using Invisible Lights. *Proceedings of the ACM Conference on Computer and Communications Security*, pages 1930–1944, 2021.

[306] Y. Wang, Y. Wang, H. Qin, H. Ji, Y. Zhang, and J. Wang. A Systematic Risk Assessment Framework of Automotive Cybersecurity. *Automotive Innovation*, 4(3):253–261, Aug 2021.

[307] B. T. Weinmann, R.P Schmotzle. Tbone. 2021.

[308] N. Weiss, M. Schrötter, and R. Hackenberg. On threat analysis and risk estimation of automotive ransomware. In *Proceedings of the 3rd ACM Computer Science in Cars Symposium*, CSCS '19, New York, NY, USA, 2019. Association for Computing Machinery.

[309] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. K. Pontes, D. Ramanan, P. Carr, and J. Hays. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks 2021)*, 2021.

[310] J. Wiseman. Gps interference map for Põlva, Estonia on january 10, 2025. `https://gpsjam.org/?lat=58.84802&lon=26.74200&z=5.2&date=2025-01-10`, note = Accessed: 2025-04-15, 2025.

[311] M. Wolf, A. Weimerskirch, and C. Paar. Security in automotive bus systems. In *Proceedings of the workshop on Embedded Security in Cars (ESCAR)'04*, 2004.

[312] C. Wolschke, S. Marksteiner, T. Braun, and M. Wolf. An agnostic domain specific language for implementing attacks in an automotive use case. In *Proceedings of the 16th International Conference on Availability, Reliability and Security (ARES)*. ACM Press, 2021.

[313] L. Wouters, E. Marin, T. Ashur, B. Gierlichs, and B. Preneel. Fast, furious and insecure: Passive keyless entry and start systems in modern supercars. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019, Issue 3:66–85, 2019.

[314] S. Wright. Autonomous cars generate more than 300 tb of data per year. Tech Blog, Tuxera, Finland, 2021.

[315] P. Wu, X. Jia, L. Chen, J. Yan, H. Li, and Y. Qiao. Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline. In *NeurIPS*, 2022.

[316] Z. Wu, S. N. Lim, L. S. Davis, and T. Goldstein. Making an Invisibility Cloak: Real World Adversarial Attacks on Object Detectors. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12349 LNCS:1–17, 2020.

[317] X. Zhu, X. Li, J. Li, Z. Wang, and X. Hu. Fooling thermal infrared pedestrian detectors in real world using small bulbs. In *in AAAI*, 2021.

[318] S. Xia, M. Qiu, M. Liu, M. Zhong, and H. Zhao. AI Enhanced Automatic Response System for Resisting Network Threats. In M. Qiu, editor, *Smart Computing and Communication*, pages 221–230, Cham, 2019. Springer International Publishing.

[319] C. Xiao, D. Yang, B. Li, J. Deng, and M. Liu. MeshAdv: Adversarial meshes for visual recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:6891–6900, 2019.

[320] C. Xu, W. Ding, W. Lyu, Z. Liu, S. Wang, Y. He, H. Hu, D. Zhao, and B. Li. Safebench: A benchmarking platform for safety evaluation of autonomous vehicles, 2022.

[321] K. Xu, G. Zhang, S. Liu, Q. Fan, M. Sun, H. Chen, P. Y. Chen, Y. Wang, and X. Lin. Adversarial T-Shirt! Evading Person Detectors in a Physical World. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12350 LNCS:665–681, 2020.

[322] W. Xu, H. Zhou, N. Cheng, F. Lyu, W. Shi, J. Chen, and X. Shen. Internet of vehicles in big data era. *IEEE/CAA Journal of Automatica Sinica*, 5(1):19–35, 2018.

[323] Y. Cao, N. Wang, C. Xiao, D. Yang, J. Fang, R. Yang, Q. A. Chen, M. Liu, and B. Li. Invisible for both camera and lidar: Security of multi-sensor fusion based perception in autonomous driving under physical-world attacks. In *IEEE S&P, IEEE*, 2021.

[324] C. Yan, W. Xu, and J. Liu. Can you trust autonomous vehicles: Contactless attacks against sensors of self-driving vehicle. *Def Con*, 2016.

[325] K. Yang, T. Tsai, H. Yu, M. Panoff, T.-Y. Ho, and Y. Jin. Robust roadside physical adversarial attack against deep learning in lidar perception modules. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, ASIA CCS '21, page 349–362, New York, NY, USA, 2021. Association for Computing Machinery.

[326] T. Yarygina and C. Otterstad. A Game of Microservices: Automated Intrusion Response. In S. Bonomi and E. Rivière, editors, *Distributed Applications and Interoperable Systems*, pages 169–177, Cham, 2018. Springer International Publishing.

[327] T. Yoshizawa, D. Singelée, J. T. Muehlberg, S. Delbruel, A. Taherkordi, D. Hughes, and B. Preneel. A survey of security and privacy issues in v2x communication systems. *ACM Comput. Surv.*, 55(9), Jan. 2023.

[328] M. Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 2000.

[329] D. Zelle, R. Rieke, C. Plappert, C. Krauß, D. Levshun, and A. Chechulin. Sepad – security evaluation platform for autonomous driving. In *2020 28th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pages 413–420, 2020.

[330] W. Zeng, M. Wu, P. Chen, Z. Cao, and S. Xie. Review of shared online hailing and autonomous taxi services. *Transportmetrica B: Transport Dynamics*, 11(1):486–509, 2023.

[331] F. Zhang, H. A. D. E. Kodituwakku, J. W. Hines, and J. Coble. Multilayer data-driven cyber-attack detection system for industrial control systems based on network, system, and process data. *IEEE Transactions on Industrial Informatics*, 15(7):4362–4369, 2019.

[332] K. Z. Zhang. Applications and prospects of ai in autonomous cars - take tesla as an example. In *2nd International Conference on Mechatronic Automation and Electrical Engineering (ICMAEE 2024)*, volume 2024, pages 355–360, 2024.

[333] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid. Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, ASE '18, page 132–142, New York, NY, USA, 2018. Association for Computing Machinery.

[334] Q. Zhang, S. Hu, J. Sun, Q. A. Chen, and Z. M. Mao. On adversarial robustness of trajectory prediction for autonomous vehicles. *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*.

[335] Y. Zhang, H. Foroosh, P. David, and B. Gong. CAMOU: Learning physical vehicle camouflages to adversarially attack detectors in the wild. *The International Conference on Learning Representations (ICLR)*, 2019.

[336] Y. Zhang and K. Rasmussen. Detection of electromagnetic interference attacks on sensor systems. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 203–216, 2020.

[337] Y. Zhao, H. Zhu, R. Liang, Q. Shen, S. Zhang, and K. Chen. Seeing isn't believing: Towards more robust adversarial attack against real world object detectors. *Proceedings of the ACM Conference on Computer and Communications Security*, pages 1989–2004, 2019.

[338] Zhisheng Hu, Junjie Shen, Shengjian Guo, Xinyang Zhang, Zhenyu Zhong , Qi Alfred Chen, Kang Li. Pass: A system-driven evaluation platform for autonomous driving safety and security. In *Fourth International Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*, 2022.

[339] Z. Zhong, G. Kaiser, and B. Ray. Neural network guided evolutionary fuzzing for finding traffic violations of autonomous vehicles. *IEEE Trans. Softw. Eng.*, 49(4):1860–1875, Apr. 2023.

[340] S. D. E. C. L. Zhongyuan Hau, Kenneth T Co. Object removal attacks on lidar-based 3d object detectors. In *Third International Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*, 2021.

[341] Y. Zhu, C. Miao, T. Zheng, F. Hajiaghajani, L. Su, and C. Qiao. Can We Use Arbitrary Objects to Attack LiDAR Perception in Autonomous Driving? *Proceedings of the ACM Conference on Computer and Communications Security*, pages 1945–1960, 2021.

[342] S. A. Zonouz, H. Khurana, W. H. Sanders, and T. M. Yardley. RRE: A Game-Theoretic Intrusion Response and Recovery Engine. *IEEE Transactions on Parallel and Distributed Systems*, 25(2):395–406, 2014.

[343] Zoox. The zoox robotaxi rolls into san francisco. `https://zoox.com/journal/zoox-robotaxi-in-san-francisco`, Nov. 2024. Accessed: 2025-01-15.

List of Figures

List of Tables

Acknowledgments

This research has been financially supported by European commission projects ECHO (No 830943), CitySCAPE (No 883321), IRIS (No 101021727), MariCybERA (Agreement No.: 952360) and also Erasmus + internship.

I would firstly like to thank my opponents Prof. Dr. Iain Phillips (Loughbrough University) and Asst. Prof. Monowar Hasan (Washington State University) for reviewing my thesis and attending the defence. I would like to thank Prof. Dr. Dirk Draheim for providing the internal review and being on the defence committee. I would like to thank Assoc. Prof. Dr. Sven Nõmm for agreeing to be the defence committee chair. I would further like to thank Kristi Ainen, Elena Vaarmets and Katri Kadakas for the administrative support to publish the thesis dissertation and organise the defence.

I would like to express my deepest gratitude to my supervisors Prof. Dr. Olaf Maennel, Dr.Ing. Mohammad Hamad and Prof. Dr. Raivo Sell. I expressly like to thank Prof. Dr. Olaf Maennel who has provided dedicated support and encouragement over the last 6 years. I would like to thank Prof. Dr. Sebastian Steinhorst and the staff and students of the Chair for Embedded Systems and Internet of Things at Technical University of Munich for hosting me for the last 2 years. I would further like to thank Dr.Ing. Mohammad Hamad for providing me the opportunity to collaborate with his research group and for being a great role model for young cybersecurity academics. Special thanks to Mohsen Malayjerdi and Mauro Bellone for the academic collaboration and their consistent support of my research. I would further like to thank Prof. Dr. Rain Ottis for his continual support and guidance throughout my PhD and career at TalTech.

Lastly, I would like to thank my family, my uncle Brett for always supporting me, my mother for her unconditional love and my wife Vanessa for always being there for me, for being the sunshine in my day and her unconditional love.

Abstract

Cybersecurity Testing and Attack Propagation Analysis of Autonomous Driving Software

Autonomous driving software needs to be robust and resilient to cyber attacks to ensure the safety of passengers and road users. Software for highly automated vehicles in which driving actions are taken and supervised by software, are in developmental stage. Software developers and control system designers must contend with the complexity of massive parallel real-time system operations in a densely connected system-of-system environment. As the software architecture of autonomous driving is developing, there are a preponderance of challenges for cybersecurity. Software developers and control system designers require an understanding as to how cyber attacks discernibly propagate throughout the autonomous driving architecture and affect the decision-making of the vehicle. Additionally, there is a need to explore methods for fingerprinting the effects of cyber attacks and debugging failures of the autonomy caused by these attacks to pinpoint vulnerabilities within the software. As autonomous driving systems are a dynamic, real-time system, it is imperative to explore options for intrusion response to cyber attacks, to mitigate or deter risks to safety. Furthermore, the development of testing tools to facilitate agile and repeatable testing is of great importance. The objective of this thesis is to develop new methods for cybersecurity testing of autonomous driving software. Therefore, new approaches to testing and evaluation, debugging, intrusion response and design of testing tools.

The thesis starts by developing a combined safety and cybersecurity testing methodology. The methodology incorporates safety metrics (distance-to-collision, acceleration, braking, steering etc.), parameters for cybersecurity (attack weighting/density etc.) and safety validation analysis to discern the effect to the autonomous driving software of cyber attacks. Utilising this approach to conduct experiments using a testing tool-chain, consisting of a digital-twin simulation testbed and a real-world testbed, vulnerabilities of the planning module for navigation and the localisation module used in a real-world autonomous driving system were found. Scenario-based testing, which focuses on an overtaking scenario, revealed the planning module was vulnerable to sensor manipulation attacks of the LiDAR and localisation sensors during the cut-in process, where the target vehicle is executing the overtake of the passing vehicle. An attack triggered during the cut-in induces decision-making uncertainty which results in erratic, attempted overtaking and side collision to the passing vehicle. Electromagnetic interference attacks were also conducted within a purpose-built hybrid testbed environment consisting of actuation processes and the high-level autonomous driving software. The evaluation of the electromagnetic interference attacks demonstrated that an attack on the steering actuation sensor, with minimal noise, could propagate through the software architecture and exploit weakness in the sub-modules of the path planning software, consisting of trajectory generation and waypoint following. Attacks to the localisation software which were conducted in a digital-twin simulation environment and also included a dataset from real-world GPS spoofing against an operational autonomous vehicle shuttle operating in the city, revealed vulnerabilities in the design of the localisation module. During a GPS spoofing attack the autonomous vehicle shuttle lost localisation, and the localisation module was unable to hold a position in relative proximity on the map and this resulted in the re-plotting of sub-optimal and unsafe trajectories. From this attack, the thesis developed, ADAssure, a methodology to debug autonomous driving software that utilises backstepping to pinpoint the root cause of failure. The ADAssure method comprises analysing

the vehicle dynamics data (steering angel, yaw angle, yaw derivative, acceleration etc.) in comparison with sensing data, to develop assertions of the system under attack. The evaluation, using diverse localisation attacks, found three assertions consisting of displacement of yaw angle within a time threshold that challenges the physical limitations of the steering model, multiple trajectory transitions with a steering angle of 180 degrees and deviation of altitude and position co-variance which result in a spike in velocity. These assertions can be used within the domain application of an autonomous vehicle shuttle for public transportation, to detect vehicle dynamic changes characteristic of cyber activity. The thesis introduces REACT, a proposed architecture for intrusion response in automotive systems. REACT contains methodology for response evaluation, and various response selection methods. We evaluate REACT on two diverse attack cases of an adversarial sample targeted at the camera sensor and information disclosure of the infotainment system. The results demonstrates that the algorithms used for optimal selection of response had sub-optimal performance for automated intrusion response in automotive, however, presented encouraging results for proposing follow-up responses to vehicle security operations centre for further action. The thesis further contributes tools for autonomous driving cybersecurity testing. FuzzSense and ADSecLang are an initial proof-of-concept tools for fuzzing and structured cybersecurity testing. FuzzSense is a conceptual architecture for fuzzing diverse layers of the autonomous driving software, the simulator, the driving scenario and the sensor data. ADSecLang provides a domain specific language for cybersecurity testing of autonomous driving software. The results of the experimentation for FuzzSense, fuzzing LiDAR sensing data, found vulnerabilities in the Autoware.Universe software. ADSecLang developed cyber attack scenarios for manipulation of camera sensing which revealed vulnerabilities in the camera-sensing based perception module designed for Apollo software. To further contribute to the community, the data of all of the experiments conducted in this research are made available in ADSecData Platform, a conceptual community data sharing environment developed in this thesis to enhance autonomous driving cybersecurity testing and product development. This thesis contributes numerous diverse testing methods and validates their utility through experimentation on a real-world, operational vehicle.

Kokkuvõte

Autonoomse sõiduki juhtimistarkvara küberturvalisuse testimine ja rünnakute leviku analüüs

Autonoomse sõiduki juhtimistarkvara peab olema vastupidav küberrünnakutele, et tagada reisijate ja liiklejate ohutus. Tarkvara autonoomsetele sõidukitele, mille juhtimistoimin-guid teostab ja kontrollib tarkvara, on arendusjärgus. Insenerid peavad mõistma, kuidas küberrünnakud mõjutavad autonoomsete sõidukite tarkvara. Kuna autonoomsete sõidukite tarkvara arhitektuur on arengujärgus, siis on palju küberturvalisusega seotud väljakutsed. Tarkvaraarendajad ja juhtimissüsteemide disainerid peavad mõistma, kuidas küberrünnakutega kaasnevad mõjud levivad üle kogu arhitektuuri ja mõjutavad sõiduki otsuste tegemist. Lisaks, on vaja uurida meetodeid küberrünnakute mõjude hindamiseks ja nendest rünnakutest põhjustatud tõrgetest taastumiseks, et teha kindlaks tarkvara haavatavused. Kuna autonoomne sõiduki juhtimistarkvara on dünaamiline reaalajas tegutsev süsteem, on hädavajalik uurida võimalusi küberrünnakutele reageerimiseks, et leevendada või ära hoida ohutusega seotud riske. Lisaks on väga oluline testimisvahendite väljatöötamine, et hõlbustada testimist. Käesoleva lõputöö eesmärgiks on välja töötada uued meetodid autonoomse sõiduki juhtimistarkvara küberturvalisuse testimiseks. See sisaldab uusi lähenemisviise testimisele ja hindamisele, rünnakutele reageerimisele ja testimisvahendite disainile. Lõputöö algab kombineeritud ohutuse ja küberturvalisuse testimise metoodika väljatöötamisest. Metoodika sisaldab ohutusmõõdikuid, küberturvalisuse parameetreid ja ohutuse valideerimise analüüsi, et kindlaks teha küberrünnakute mõju autonoomse sõiduki juhtimistarkvarale. Kasutades seda lähenemisviisi eksperimentide läbiviimiseks testimisprotsessis, mis koosneb digitaalse kaksiku simulatsiooni keskkonnast ja reaalse maailma simulatsiooni keskkonnast, leiti haavatavused planeerimismoodulis ja lokaliseerimismoodulis. Stsenaariumipõhine testimine, mis keskendub möödasõidu stsenaariumile, näitas, et planeerimismoodul oli haavatav anduri manipulatsioonile. Anduri manipulatsioon põhjustab autonoomse sõiduki avarii. EMI rünnakuid testiti ka hübriid-testkeskkonnas. EMI rünnakute analüüs näitas, et planeerimismoodulil olid haavatavused. Rünnakud lokaliseerimistarkvarale, paljastasid haavatavused lokaliseerimismooduli disainis. GPS-signaali segamise rünnaku ajal kaotas autonoomne sõiduk GPS asukoha andmed. Sellest rünnakust lähtudes töötati välja ADAssure metoodika, mis kasutab tõrke algpõhjuse väljaselgitamiseks "alt üles" meetodit. ADAssure'i meetod hõlmab sõiduki andmete analüüsi, et töötada välja reeglid rünnatava süsteemi kohta. GPS-signaali segamise rünnaku analüüsi kaudu leiti 3 reeglit. Reegleid saab kasutada sõiduki turvalisemaks muutmiseks. Lõputöö tutvustab REACT-i - arhitektuuri küberintsidentidele reageerimiseks. REACT sisaldab küberintsidentidele reageerimise vastuse hindamise metoodikat. Hindame REACT-i, kasutades kahte rünnaku stsenaariumi. Tulemused näitavad, et REACT-i arhitektuuri kasutamine on kasulik. Lõputöö tutvustab kahte tööriista - FuzzSense ja ADSecLang - küberturvalisuse testimiseks. See lõputöö annab panuse arvukatesse erinevatesse testimismeetoditesse ja kinnitab nende kasulikkust.

Appendix I

Paper I

A. Roberts, L. Teply, M. Bellone, M.Pese, O. Maennel, M. Hamad, and S. Steinhorst. Fuzzsense: Towards a modular fuzzing framework for autonomous driving software. In arXiv, 2025.

arXiv:2504.10717v1 [cs.CR] 14 Apr 2025

FuzzSense: Towards A Modular Fuzzing Framework for Autonomous Driving Software

Andrew Roberts*
* FinEst Centre for Smart Cities,
Tallinn University of Technology

Lorenz Teply, Mohammad Hamad, Sebastian Steinhorst†
†Department of Computer Engineering,
Technical University of Munich

Mert D. Pesé‡
‡School of Computing,
Clemson University

Olaf Maennel§
§School of Computer and Mathematical Sciences,
The University of Adelaide

***Abstract*—Fuzz testing to find semantic control vulnerabilities is an essential activity to evaluate the robustness of autonomous driving (AD) software. Whilst there is a preponderance of disparate fuzzing tools that target different parts of the test environment, such as the scenario, sensors, and vehicle dynamics, there is a lack of fuzzing strategies that ensemble these fuzzers to enable concurrent fuzzing, utilizing diverse techniques and targets. This research proposes *FuzzSense*, a modular, black-box, mutation-based fuzzing framework that is architected to ensemble diverse AD fuzzing tools. To validate the utility of FuzzSense, a LiDAR sensor fuzzer was developed as a plug-in, and the fuzzer was implemented in the new AD simulation platform AWSIM and Autoware.Universe AD software platform. The results demonstrated that FuzzSense was able to find vulnerabilities in the new Autoware.Universe software. We contribute to FuzzSense open-source with the aim of initiating a conversation in the community on the design of AD-specific fuzzers and the establishment of a community fuzzing framework to better target the diverse technology base of autonomous vehicles.**

I. Introduction

Fuzz testing of autonomous driving (AD) software aims to use unsanitized and invalid input to trigger exceptional or abnormal behavior of the driving logic. AD fuzzers are designed in a disparate manner, seeding input from either the sensor data, vehicle dynamics data, scenario and simulator configuration. EnFuzz [1] demonstrated that a collective framework could ensemble diverse fuzzers exhibiting different fuzzing techniques to obtain deeper penetration of one specific type of target, in this instance, application software. As the architecture of AD software relies on a mixture of different sensor technologies and data sources, the innovation required of ensemble fuzzing for AD software is that the framework must be extensible to allow different fuzzers for different targets and target groups. Our idea with this research is to explore such a concept as an ensemble fuzzing framework for AD software and present our ideas on how this could be architected. To this end, we present *FuzzSense* (Fig. 1), a conceptual architecture based on a modular, black-box, mutation-based fuzzing framework.

The architecture of FuzzSense is envisioned to integrate within the AD software simulation environment (CARLA, AWSIM, Apollo), allowing diverse fuzzing tools as plug-ins to generate test cases, collect output data in a seed corpus, and mutate new inputs. Our motivation in presenting this work is to provoke discussion within the community on how AD systems are fuzzed, establish community efforts for fuzzing and to gather initial feedback on FuzzSense and understand potential improvements on the foundations of the design of the framework. This work is not a benchmarking study or a statistical evaluation of fuzzing performance, as the motivation is purely to understand how the design of an overarching fuzzing framework for AD software may be achieved. Therefore, to clearly state the contributions of this work, we have focused on the development of the initial concept of the AD ensemble fuzzing framework, developed source code, and conducted an initial test case.

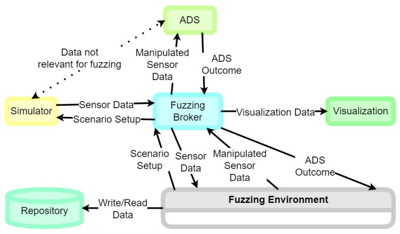


Fig. 1. High-level Architecture all Components

At a more detailed level, our contributions are listed as the following:

- We present FuzzSense, an ensemble fuzzing framework for AD software.
- We develop a sensor fuzzing plugin for FuzzSense for the LiDAR sensor from reverse engineering LiDAR sensor configurations and applied it within AWSIM, which uses more advanced LiDAR representation technology (Rings) than popularly used CARLA.
- We demonstrate an initial test case of a FuzzSense plug-in to find vulnerabilities in state-of-the-art Autoware.Universe software.
- We provide FuzzSense open-source to the community to utilize in fuzzing testing/research [FuzzSense].

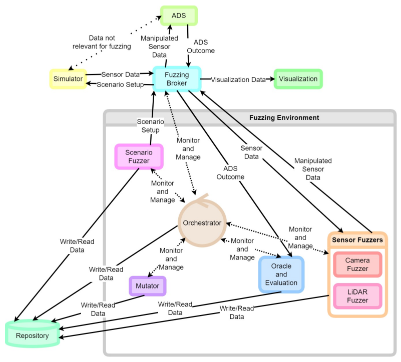


Fig. 2. FuzzSense: High-level Architecture of Fuzzing Framework

II. FuzzSense

FuzzSense involves the following key components: the fuzzing broker, the fuzzing environment, and the repository. The interactions of these key components with the ADS and simulator are displayed in Fig. 2.

A. *Fuzzing Broker*

The Fuzzing Broker is the central part of the FuzzSense framework, acting as an intermediary layer, facilitating communication between the simulator, ADS, and fuzzing environment. The fuzzing broker has full control over the exchanged sensor data and listens to data, such as steering commands. While the Fuzzing broker was described as an intermediary for the whole framework, it additionally functions as a controller, initiating and terminating the operations in the connected Simulator and ADS. Depending on the used Fuzzers, Simulator, and ADS, the Fuzzing Broker transforms the sensor data to the required formats of the endpoints.

B. *Fuzzing Environment*

The Fuzzing Environment is the collection of the components responsible for fuzzing and creating scenarios, manipulating the sensor data, interpreting the results, and mutating parameters. This continues the idea of the modular architecture of the fuzzing framework. It also allows for the decomposition of other integrated modules, as the Mutator is not required to be a part of the fuzzers. The Fuzzing Environment contains the following modules: orchestrator, mutator, scenario fuzzer, sensor fuzzer/s, and oracle and evaluation.

a) Orchestrator:: The Fuzzing Environment is a composition of diverse components with unique tasks. The role of the orchestrator is to provide a central management function to ensemble these diverse components to achieve the task of fuzzing the selected targets. The idea of a fuzzing orchestrator performing a central management role was inspired by EnFuzz [1], which uses a similar design to integrate and manage diverse fuzzing modules using diverse techniques. The Orchestrator possesses the intelligence in the Fuzzing environment. This is reached by always knowing the current status of the fuzzing campaign and its iterations, therefore, it can start fuzzing iterations, telling each component (Fuzzers, Oracle, Mutators, Fuzzing Broker) when they should perform which of their tasks, monitor the components to understand their status to be able to efficiently start the next step with the required components. This requires the Orchestrator to use adapters to communicate to the APIs of the different fuzzing modules. As such, no inter-communication is required for different fuzzing modules; hence, this communication is managed centrally by the Orchestrator. The benefit of central management is that expected new fuzzing modules can be integrated in less time and with less complexity. Further, it even allows decoupling the mutation of parameters and the fuzzers where they are processed.

b) Mutator:: The Mutator creates the parameters utilized by the scenario and sensor fuzzing modules. In the first round/s the Mutator is providing the fuzzers with the seeds but does no actual mutation on them. In this architecture, the Mutator is extracted from the scenario and sensor fuzzers. The aim is to allow the combination of different mutation algorithms and fuzzers. Furthermore, it allows a closer synchronization between the mutation of parameters when using multiple fuzzers. For the proof of concept, the mutation is a brute-force/grid search iteration through parameters, where limits are applied and derived from logical boundaries like the perception distance of the sensors.

c) Scenario Fuzzer:: Scenario fuzzers use parameters of the driving scenario as the seed pool. These can include weather, pedestrians, and other vehicles. Mutations can be built from the mission, weather, and scenario actors. Prominent scenario fuzzers include only the distinct module creating the scenarios based on parameters given by the Mutator, which is called the Scenario Fuzzer in the FuzzSense architecture.

d) Sensor Data Fuzzers:: Autonomous vehicles can use a range of sensor technologies and different hardware and software configurations and can be positioned at different locations on the vehicle. In general, the sensor data of any of those sensors could be fuzzed. The idea motivating our ensemble fuzzing design is that a dedicated sensor data fuzzing plug-in is responsible for each sensor data stream that should be fuzzed. The parameters provided by the Mutator can either be synchronized between several or be mutated individually.

e) Oracle and Evaluation:: The Oracle and the Evaluation are giving further intelligence to the Fuzzing Environment. The Oracle and Evaluation component is responsible for creating ground truth, known commonly as the Golden Run. Afterward, every fuzzing iteration must be checked for possible findings, and thus, the Mutator must be provided with an evaluation of the parameters. This framework does not suggest certain conditions once a finding is detected. The idea is to set this based on the subject of testing. For instance, it could be limited to deviations of the trajectory of the Golden Run or only focus on temporal aspects (speed of the vehicle, etc.) introduced by the fuzzing.

C. Repository

In this architecture the repository enables the Fuzzing Environment to write logs, store data and dependent on the communication allow the components to exchange data. When the modules exchange data using the repository, it allows a decoupling and a simpler integration of new components, especially, because the orchestrator is handling the management centrally and thus modules do not need custom integrations with all other required in the Fuzzing Environment.

III. Sensor Data Fuzzing

AD software relies on sensing data for situational awareness and to inform navigation and motion-planning activities. *FuzzSense* is designed to apply manipulations to the sensor data stream before it reaches the downstream AD software. The initial version of the fuzzer manipulates pixels in the camera feed or points in the LiDAR feed. The fuzzer is triggered during a scenario simulation. For each future scenario, the fuzzing test case is mutated based on evaluation of the feedback. The delivery of the manipulation of the sensor data is achieved through the application of changing or adding data in the data stream based on the coordinates given by the *fuzzing mask*.

A. Fuzzing Mask

The *fuzzing mask* is created based on parameters given by the sensors and vehicle that are to be tested. For the camera stream, which can be represented as a matrix with definitions of each pixel's coordinates, color, and sometimes the alpha channel, the *fuzzing mask* provides a collection of coordinates for pixels that are changed in the camera stream. For LiDAR, the same concept is used to add points to the point cloud, and only the distance is added. Our goal is to achieve several advantages with this approach. First, the same mutation strategy for most parameters can be used. Second, the computation of the next data points to manipulate in the LiDAR data stream is independent of the actual point cloud data. This could potentially increase the performance. Third, by limiting the space of possible manipulations in the search space, possible mutations of the parameters can be drastically reduced to the areas of interest (e.g., in front of the vehicle). Within a point cloud, points can be hidden behind others from the sensors perspective. The concept with the *fuzzing mask* prevents such cases so that no added points are shadowed by other added points (see Fig. 3).

The fuzzing mask $\mathcal{F}$(Algorithm 1) is defined as a set of coordinates where the sensor data is manipulated $\mathcal{F} = \{(x_i, y_i) \mid x_i \in [0, W], y_i \in [0, H]\}$. For the camera sensor, the location of the pixel, and for the LiDAR sensor corresponds to the coordinates of a perpendicular plane in the pointcloud where each point is inserted. The third dimension for LiDAR is provided by the distance parameter between the LiDAR sensor and the plane. The coordinates are relative to width, height, and, for 3D data, the center of the plane. For the camera stream, they are taken from the metadata of the sensor stream, and for LiDAR, they are preset and could potentially be mutated.

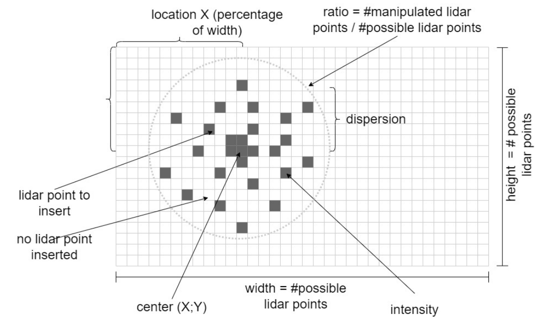

location X (percentage of width)
ratio = #manipulated lidar points / #possible lidar points
dispersion
height = # possible lidar points
lidar point to insert
no lidar point inserted
width = #possible lidar points
center (X;Y)
intensity

Fig. 3. Fuzzing Mask for LiDAR.

Algorithm 1 Generate Fuzzing Mask $\mathcal{F}$

```
Require: $r_f, \sigma_f, X, Y, W, H$
1: $(\sigma_x, \sigma_y) \leftarrow (W * \sigma_f, H * \sigma_f)$
2: $r_f \leftarrow W * H * r_f$
3: $\overline{x} = \mathcal{N}(r_f, \sigma_x, X, W)$
4: $\overline{y} = \mathcal{N}(r_f, \sigma_y, Y, H)$
5: for $i \leftarrow 0$ to $r_f - 1$ do
6:     $\mathcal{F} \leftarrow add(x[i], y[i])$
7: end for
8: return $\mathcal{F}$
```

Let r_f represent the fuzzing change ratio, defined as $r_f = \frac{N_c}{W \times H}$. Where: N_c is the number of changed data points, W and H are the width and height of the fuzzing mask matrix in discrete steps (e.g. pixels for the camera stream). The result is expressed as a percentage. Then, let σ_f represent the standard deviation of the manipulated data points, computed as the deviation relative to width W and height H. Together, X and Y are the coordinates of the center of the fuzzing mask and the means of the standard-deviation. $\overline{x}$ and $\overline{y}$ are the vectors corresponding to the each x and y coordinate vector respectively. In line 3-4 W and H ensure, that no coordinates outside of the fuzzing mask are created. Where in line 6 $\mathcal{F}$ is created by column stacking the $\overline{x}$ and $\overline{y}$ arrays with the calculated normal-distributions.

B. Multi-Stage Approach

FuzzSense combines multiple stages during fuzzing. Each time the fuzzing setup is started, it is called a **Fuzzing Campaign**. Each of the scenarios running with different fuzzing parameters is defined as a **Fuzzing Iteration**. This allows to better distinct between phases and to have an easier understanding of the complete process and architecture. The aim of this process design choice is that the focus for the fuzzing campaign can be chosen with more granularity as the multi-stages allows to provide intelligence to the iterations. The logic when to exit the inner iteration (sensor fuzzing iteration) can be set based on the aim of the fuzzing campaign. This is possible, because the inner and outer iteration (scenario fuzzing iteration) can be logically separated.

1) Fuzzing Campaign: The Fuzzing Campaign defines the whole duration of the fuzzer running. A Fuzzing Campaign consists of one or many Fuzzing Iterations. To start a fuzzing campaign, one or several seeds are required. Each seed contains starting values for each parameter. While there is not any condition met, which qualifies the end of the campaign, new scenario fuzzing iterations are started. The campaign also could be stopped manually. The final step is to stop all required services and store the results from the fuzzing campaign to allow further investigations.

2) Fuzzing Iteration: The Fuzzing Iteration defines one single scenario run. It starts with the parameter mutation and ends once the scenario is stopped because of a failure or because it has successfully finished. The fuzzing of every single data frame is not called iteration. A here defined Fuzzing Iteration includes all those manipulated sensor data frames throughout the whole scenario until it finishes or fails with a finding. As the main focus of the fuzzing is on the sensor data, the mutation for the scenario parameters is not performed in every iteration. Thus, the same scenario is present throughout several iterations. To distinguish also between those two, there can be *Scenario Fuzzing Iterations* and *Sensor Fuzzing Iterations*. One Scenario Fuzzing Iteration consists of one or many Sensor Fuzzing Iterations.

a) Scenario Fuzzing Iteration: The ADS of the AV must act within a scenario to allow relations to its intended real-world use. A scenario defines not only the ego-vehicle itself but also the road, traffic signs, and signals, road conditions, environment, other actors, including their behavior, and the weather conditions. The Scenario Fuzzing Iteration is the outer iteration and contains all Sensor Fuzzing Iterations in the same scenario. It contains the following steps:

Step 1:Mutate Scenario Parameters
Step 2:Create a Scenario and set it up in the simulator and ADS
Step 3:Create Golden Run
Step 4:Start *Sensor Fuzzing Iterations*

b) Sensor Fuzzing Iteration: Within the same Scenario Fuzzing Iteration, the parameters for the Fuzzing Mask should not be the same twice. However, within a new Scenario Fuzzing Iteration, the same parameters can be used again. Each sensor fuzzer takes the original sensor data from the simulator and applies manipulations to the data stream before it reaches the ADS. Those manipulations are single pixels in the camera feed or points in the LiDAR feed. In the current state, within one run, the planned drive of the vehicle, no mutations on the parameters are performed. This means the same fuzzing masks are applied to the data streams from the start to the end of the drive. The mutator is only active between runs. Therefore, compared to a plain simulation, the only computational overhead during a running simulation is the rerouting and manipulation of the sensor data. It contains the following steps:

Step 1:Mutate Sensor Parameters
Step 2:Set scenario up in simulator and ADS
Step 3:Create Fuzzing Masks
Step 4:Start scenario and manipulate sensor data streams

IV. Experiment & Results

A. Experimental Setup

The evaluation of FuzzSense is conducted in AWSIM, a high-fidelity, digital-twin simulation environment. The target AD system uses the Autoware.Universe software framework. As this instantiation of the AD software uses the LiDAR sensor for perception and localisation, the sensor fuzzing module is configured to fuzz the LiDAR sensor. The evaluation was conducted on a system running Ubuntu 22.04.03 LTS with 1 TB of storage, 32 GB of CPU memory, 10 GB of GPU memory, a 12th Gen Intel® Core™ i7-12700KF processor, and a GeForce RTX 3080 Lite Hash Rate graphics card.

B. Results & Discussion

The driving scenario consists of a planned navigation in an urban driving environment. We selected an urban environment since attacks can cause more severe effects within a congested operational driving domain. As the vehicle navigates through its planned trajectory, the sensor fuzzing plug-in of FuzzSense initiates its fuzzing mask, manipulating the parameters of the LiDAR 3D geometry. For this set of experiments, the parameters were randomly set at x (0.4),y (0.5), the distance of the fuzzed LiDAR points (30m), and the intensity 0.1. and dispersion (width 100, height 60). The experiments mutated the location and dispersion parameters. The fuzzing broker is fuzzing every frame. In the simulation, the environment exhibits a performance of time of approx. 30 frames per second or 33 milliseconds. Fig. 4 displays the initiation of the fuzzing mask (the yellow box is used for identification and does not represent the full mask) to the driving simulation. The fuzzing mask is applied at different distances from the vehicle and different locations within the environment. As shown in Fig. 4, the fuzzing mask is located at an approaching distance to the vehicle of approx. 30 meters outside the lane does not produce any unsafe changes in the vehicle the vehicle's behavior.

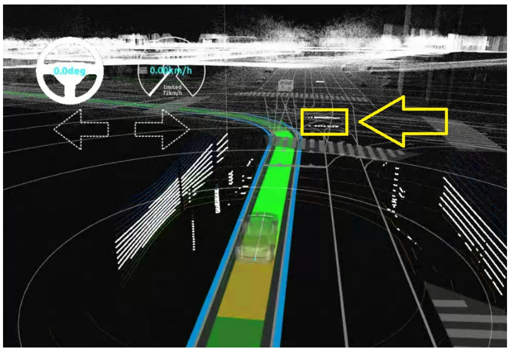


Fig. 4. Fuzzing Mask applied to the right edge of lane

Fig. 5 displays the movement of the fuzzing mask to a more central location in the driving environment. The fuzzing parameters for amount and dispersion are the same as Fig. 4 in both fuzzing iterations. The parameter for the distance is the same for both. The affect of the fuzzing mask displayed

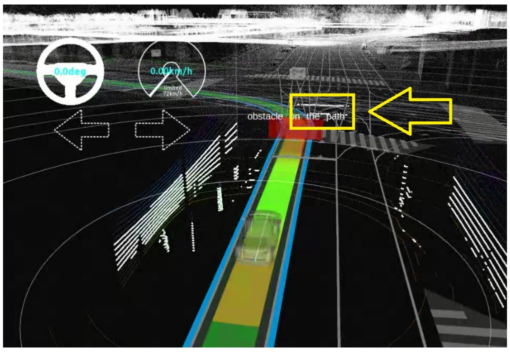


Fig. 5. Fuzzing Mask applied to central location of vehicle trajectory

in Fig. 5, is that the vehicle detects the fuzzed LiDAR points as an obstacle (red wall) and plans a reduction in acceleration to observe the obstacle. This can be seen by the orange color in the planned trajectory.

Fig. 6 displays the fuzzing mask applied at a close distance and within the planned trajectory of the vehicle. The vehicle detects the fuzzing mask as an object in immediate proximity to the vehicle and therefore initiates a braking action. The vehicle is unable to recompute an alternative planned trajectory due to the fuzzed points presenting an obstacle across the road and therefore the vehicle is unable to progress.

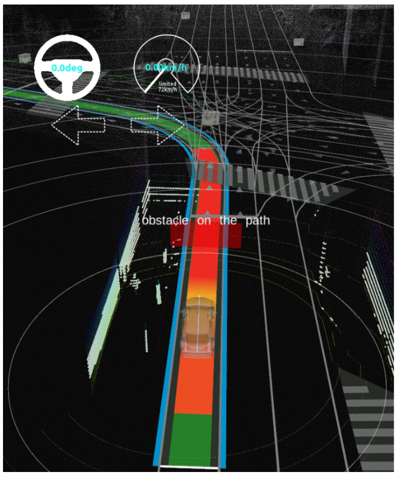


Fig. 6. Top down view of vehicle with fuzzing mask affecting planned navigation of the vehicle

The experiments provide initial feedback on the utility of FuzzSense. From observing the behaviour of the AD software, displayed in Figures 5 and 6 we can discern that sensor fuzzing is a useful exercise to find vulnerabilities of the AD software stack. The results indicate that the AD software is either unstable or can be influenced by inserted LiDAR points. We found that when the fuzzing mask was located on or near the planned trajectory of the vehicle, the perception algorithm was unable to filter the manipulated points and instead, observed them as an obstacle. Further to this, when the fuzzing mask was located in close proximity to the vehicle, it resulted in a complete stop of the vehicle.

V. Related Work

The EnFuzz architecture [1] demonstrates the advantage of combining multiple fuzzers which use diverse techniques of fuzzing, to get a greater and deeper penetration of the target. The EnFuzz design further inspired our adoption within FuzzSense of an orchestrator (monitor) for coordination. Our contribution is unique from EnFuzz as our focus is specific to AD software and we incorporate in the design considerations for the diversity of AD technology and targets.

Aforementioned, there are various fuzzers focused on disparate targets of the AD system. Popularly cited fuzzing tools include DeepRoad [2], DeepTest [3] which target the camera sensor and AV-Fuzzer [4], Auto-Fuzz [5] and DriveFuzz [6] which target the driving scenario. These fuzzers are not designed to operate concurrently with different fuzzers, but focus on a seed pool limited to there target. For the optimization of the search space reduction, these fuzzing tools mainly focus on driving quality and task performance metrics as a measure to direct the mutations towards more promising scenarios where the ego-vehicle is more likely to struggle.

Our work does not aim to compete with these fuzzers nor do we seek to build on there designers. FuzzSense, is an overarching framework whose concept is based on enabling the usage of the fuzzing tools as plug-ins in an integrated fuzzing environment. A future test case would be to use DeepRoad [2] and DriveFuzz [6] within FuzzSense to understand how diverse fuzzing techniques generate bugs.

VI. Conclusion & Future Work

In this work we presented our idea for the design of an ensemble fuzzing framework for AD software, which we call FuzzSense. FuzzSense is designed for vehicles with several different sensors. The architecture consists of many different modules which perform tasks fulfilling each a crucial part in the execution of fuzzing and of the coordinating and monitoring of those diverse fuzzing modules. We developed a plug-in for fuzzing LiDAR point clouds and utilised to find vulnerabilities in Autoware.Universe AD software. Future work, aims to experiment with FuzzSense utlising the modularity to benchmark the performance of different fuzzing plug-ins. Further, advancing the design of the fuzzing mask by adding support for further sensor types. As part of providing FuzzSense open-source, we also aim to actively gather community feedback and develop the framework further.

References

[1] Y. Chen, Y. Jiang, F. Ma, J. Liang, M. Wang, C. Zhou, X. Jiao, and Z. Su, "{EnFuzz}: Ensemble fuzzing with seed synchronization among diverse fuzzers," in *28th USENIX Security Symposium (USENIX Security 19)*, pp. 1967–1983, 2019.

[2] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, "Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, ASE '18, (New York, NY, USA), p. 132–142, Association for Computing Machinery, 2018.

[3] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: automated testing of deep-neural-network-driven autonomous cars," in *Proceedings of the 40th International Conference on Software Engineering*, ICSE '18, (New York, NY, USA), p. 303–314, Association for Computing Machinery, 2018.

[4] G. Li, Y. Li, S. Jha, T. Tsai, M. Sullivan, S. K. S. Hari, Z. Kalbarczyk, and R. Iyer, "Av-fuzzer: Finding safety violations in autonomous driving systems," in *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, pp. 25–36, 2020.

[5] Z. Zhong, G. Kaiser, and B. Ray, "Neural network guided evolutionary fuzzing for finding traffic violations of autonomous vehicles," *IEEE Trans. Softw. Eng.*, vol. 49, p. 1860–1875, Apr. 2023.

[6] S. Kim, M. Liu, J. J. Rhee, Y. Jeon, Y. Kwon, and C. H. Kim, "Drivefuzz: Discovering autonomous driving bugs through driving quality-guided fuzzing," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, (New York, NY, USA), p. 1753–1767, Association for Computing Machinery, 2022.

Appendix II

Paper II

A. Roberts, M. Malayjerdi, M. Bellone, R. Sell, O. Maennel, M. Hamad, and S. Steinhorst. Adsecdata platform: An open-source data platform for autonomous driving cybersecurity. In 2025 IEEE 101st Vehicular Technology Conference (VTC2025-Spring), pages 1–7, 2025.

ADSecData Platform: An Open-Source Data Platform for Autonomous Driving Cybersecurity

Andrew Roberts[1], Mohsen Malayjerdi[1], Mauro Bellone[1], Raivo Sell[1], Olaf Maennel[3], Mohammad Hamad[2], Sebastian Steinhorst[2]
[1]Tallinn University of Technology, Estonia
[2]Technical University of Munich, Germany
[3]University of Adelaide, Australia

***Abstract*—Autonomous driving (AD) software needs to be secure, and its decision control must be robust against cyber threats. The development of cybersecurity solutions for legacy and connected vehicles has been supported by an array of open-source datasets, mainly focused on the CAN Bus protocol. There exists a lack of open-source cybersecurity data and community-driven platforms that enable fair and reproducible evaluations of AD algorithms from a cybersecurity perspective and defensive mechanisms. This study addresses this problem by conducting an in-depth analysis of the data ecosystem for AD cybersecurity and introducing an initial open-source data platform, ADSecData. ADSecData offers the community a comprehensive 4-stage method for the creation of AD cybersecurity datasets, along with an initial common dataset. We evaluate the utility of ADSecData through a case study featuring diverse malicious injection attacks, including GPS spoofing, LiDAR point-cloud manipulation, and sensor interference. The results demonstrate the viability of ADSecData in generating AD cybersecurity datasets and supporting community research and development.**

***Index Terms*—Security, Autonomous Driving**

I. Introduction

AD software must be secure, with decision control optimized to ensure robustness against cyberattacks. A key challenge in achieving this goal is the lack of open-source data specifically for AD cybersecurity. Without available data, software designers do not have an immediate understanding of the considerations for secure design required to ensure robustness against cyber threats. In contrast, there are many open-source datasets for safety validation, algorithm optimization, and sensor configuration. Popular examples include KITTI [1], Waymo [2], Baidu Apolloscape [3], Argoverse [4] and NuScenes [5]. Common datasets for safety validation have enabled platforms such as CARLA Leaderboard [6] to establish challenges to benchmark solutions for perception and trajectory planning algorithms. The problem motivation that this research confronts is that AD cybersecurity doesn't have a readily available source of open datasets available to advance research and there is a lack of guidance on how to conduct cybersecurity research to generate datasets for benchmarking.

To confront this problem, we present *ADSecData Platform*, a consolidated platform that provides open-source AD data for cybersecurity. As shown in Fig. 1, ADSecData Platform consists of a data generation process, which is the method used to generate datasets from simulation and real-world experiments. We validate the platform in a case study using the data generation method to create datasets based on an operational autonomous vehicle (AV) program. We demonstrate the utility of our open-source platform to the community in advancing cybersecurity testing to measure and improve the robustness of autonomous driving systems to cyberattacks.

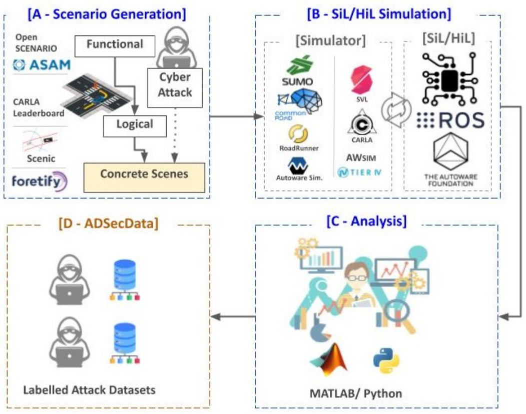


Fig. 1: ADSecData Platform - Data Generation Process.

To construct an AD cybersecurity open-source data platform, we used these guiding questions to establish an understanding of the relationship of AD data to cybersecurity:

1) What data types generated by the AD system are utilized for cyber attack test cases?
2) What is the utility of each data type in enhancing the cybersecurity of AD?
3) What metrics are available to benchmark AD algorithms from a cybersecurity perspective and defense mechanisms?

The major contributions of this research are the follows:

- We present an initial prototype of the *ADSecData Platform*, an open-source platform designed to support AD cybersecurity data.
- We offer detailed guidance on structuring cybersecurity testing frameworks to facilitate the generation of datasets for the research community.
- We contribute to the AD cybersecurity community by providing an initial common dataset and defining key challenges to focus future research and development efforts.

II. Autonomous Vehicle Cybersecurity Data

The emerging field of automotive cybersecurity research over the last decade has focused predominantly on the CAN Bus protocol, connected vehicle protocols, electrical and embedded hardware (such as wireless controllers and Bluetooth), and in-vehicle software systems (e.g., infotainment systems). To support the development of defensive technologies and the secure design of communication protocols and software, numerous open-source datasets of automotive telemetry have been created. These datasets primarily address legacy and connected vehicle technologies, with a strong emphasis on the CAN Bus protocol. However, there is a significant lack of open-source cyberattack datasets specific to AD technology. Developing such datasets and promoting the exchange of open-source data are critical steps toward advancing the still-maturing field of AD cybersecurity.

A. Autonomous Vehicle Data

AD systems generate a vast amount of data from diverse hardware and system components. We classify AD data into four major sub-categories of data sources: *sensing*, *system*, *network*, and *vehicle dynamics*. For each data source, we discuss its value for software development, cybersecurity, and its availability.

1) Sensing: Sensing data is produced by advanced sensors in the AD system, including LiDAR, cameras, ultrasonic radar, and global navigation systems (GPS, GLONASS, Baidu, Galileo). This data is critical for mapping the driving environment, perception, and localization. However, one of the key challenges with sensor data is the high data rate generated by autonomous vehicles. Xu et al. [7] estimated that diverse sensors could generate approximately four terabytes of data per day. The transmission of LiDAR and high-definition camera frames from onboard sensors to edge data logging servers further complicates data collection. Although compression techniques are available to optimize transmission efficiency, there is limited understanding of how these methods impact cybersecurity research in computer vision and perception.

Software Development Value: Sensing data is used by AD software designers to train and optimize algorithms for SLAM, object detection and tracking, sensor fusion, and semantic segmentation. One of the many examples of progress in this area is the CARLA Autonomous Driving Leaderboard [6], which is a platform used for the development of AD agents.

Cybersecurity Value: Sensing data can be used to assess vulnerabilities of AD software to adversarial examples and generate new attack models for adversarial examples. Select examples include:

- LiDAR point cloud manipulation [8]
- Adversarial examples for camera perception neural networks. [9]
- Light manipulation attacks on camera hardware and driving objects (road signs, etc.) [10]
- Fuzzing and parameter manipulation attacks against AD algorithms (Object Detection, Sensor Fusion) [11]
- GPS Spoofing causes uncertainties in trajectory planning algorithms. [12]

Defensive technologies can also be developed from sensing data, these include:

- Kalman filters and ML detection solutions to filter noise from data manipulation attacks. [13]
- Physical intrusion detection solutions which fingerprint patterns of noise from adversarial activity. [14]
- Improvements to the security of ML models to protect against ML evasion and training data poisoning attacks.

Data Availability: Open-source cybersecurity datasets for sensing, of which there are very few, predominantly focus on camera-based perception and neural networks for perception algorithms. Available datasets include:

- *Natural Denoising Diffusion Attack (NDDA) dataset* [15]
- *SlowTrack: Camera-based perception latency attack dataset* [16]

2) System: System data consists of data from the onboard software systems of the AD system. These include the firmware, operating system, application software, and real-time operating systems used in the electronic/embedded components such as the electronic control units (ECUs) and microelectronic control units (MCUs).

Software Development Value: System data is used by software developers to debug errors and understand application performance and functionality. Crucial for AV developers is to understand the performance and reliability of the AD software (Autoware, Nvidia Drive, Apollo) and middleware (Robotic Operating System (ROS), Cyber RT).

Cybersecurity Value: System data is used for vulnerability and exploit analysis. Activities that are included in this description include reverse engineering firmware, code analysis, taint-analysis, and fuzz testing.

Data Availability: System datasets are generally available from the manufacturer. These are then used for vulnerability and exploit analysis. Cybersecurity datasets are rare as the responsible disclosure process usually results in the removal and updating of new software. Examples of a cybersecurity system artifact are the following:

- *Kia OFFensivE Exploit (KOFFE) Metasploit module* [17]
- Mazda Infotainment USB attack [18]

3) Network: Network data consists of data produced from the AV internal and external networks. CAN Bus is the network that predominates in in-vehicle communication between ECUs and handles critical real-time functions such as braking and steering actuation. Automotive Ethernet is gaining in popularity and is mostly used for drive-by-wire communication. Other communication, such as MOST, is used for infotainment systems, and LIN can be found in more upmarket vehicle classes. The difficulty in providing CAN (and most other in-vehicle protocols) datasets is that CAN is used in a proprietary format by vehicle manufacturers. To decipher the meaning of CAN messages, either the manufacturer's diagnostic tool is required or knowledge to reverse engineer CAN messages from the investigation of firmware and system manuals.

For legacy and connected vehicles, great progress has been made, and there exist many available datasets and tools to help with the CAN message extraction process [19]. However, to our knowledge, no CAN cybersecurity-specific datasets exist for AD technology. Reasons for this could be the enhanced commercial sensitivity of AD technology, a more diverse range of AV manufacturers, the implementation of encrypted messaging with CAN-FD, and the cutting-edge nature of AD technology. Other network concepts typical in AD architectures include Vehicle-to-vehicle (v2v) and vehicle-to-everything (v2x), which use wireless and cellular connectivity for connectivity. Different application layer protocols are used for distinct purposes, these may include MQTT for vehicle on-board unit (OBU) to edge communication and Cooperative v2x (C-V2x) protocols that including basic safety messages (BSM) for cooperative perception and intelligent feedback for decision-making.

Cybersecurity research in this field is well-developed, and many available studies investigate attack models to the integrity of cooperative vehicular messages and the availability of networks that support vehicle data processing and cooperative communication.

Software Development Value: For software developers, network datasets can assist in understanding system interconnection and latency of data flow through situational awareness data to control actions decided by AD software and physical processes made by actuation.

Cybersecurity Value: Network datasets are primarily used for defensive intrusion detection solutions. Network datasets also aid in developing new attack strategies (DDoS, Replay, etc.) and fuzzing strategies to test the robustness of communication architectures. Lately, as more CAN cybersecurity datasets are available, research has focussed on ML and AI solutions for automated attack detection and fuzzing [20]. Within AD architectures, network data is utilized to evaluate the security aspects of cooperative driving, such as message trust and authentication. Perhaps the greatest contribution of cybersecurity CAN datasets has been the increase in attention brought by attacks, which demonstrate the feasibility of cyber attacks to manipulate safety-critical functions such as braking, steering, and acceleration. Recognition of these threats has seen the development of security within automotive software architectures (AUTOSAR Adaptive) and new zonal communication architectures for in-vehicle network communications.

Data Availability: Open-Source CAN hacking datasets exist for legacy and connected vehicles, a sample of this long list include:

- *Car-Hacking-Dataset* [21] [22]
- *Survival Analysis Dataset* [23]
- *CAN-Train-And-Test Dataset* [24] [25]
- CANet Dataset [26]
- CrySyS Dataset [27]
- *CIC IoV 2024 Dataset* [28]
- *CAN-MIRGU Dataset** [29]

**The CAN-MIRGU dataset is generated from a vehicle with AD capabilities, however, these capabilities are not detailed due to privacy reasons and the AD functions are deactivated for safety reasons.* For V2X and V2V selected datasets include:

- *Simulated VANET Attack Dataset* [30]
- *Simulated VANET Attack Dataset* [31]

4) Vehicle Dynamics: Vehicle dynamics data include body physical movement (lateral and longitudinal pose, yaw, etc.), acceleration, braking, and steering actuation. Vehicle dynamics is crucial for a software developer and cybersecurity engineer to understand how behavior at a system level affects the vehicle. Existing cyber attack research, which focuses on vehicle dynamics, predominantly concerns itself with providing artifacts such as docker images of the attack simulation and the code base for adversarial examples and fuzzing tools. A limitation of this approach is that it requires a custom configuration of the attack in the user environment and an understanding of the vehicle model and metrics engine for data output used in the original research.

Software Development Value: This data is crucial for control algorithm designers to assess the robustness of control and trajectory planning algorithms. Software developers and control designers will use vehicle dynamics data for backstepping and back-propagation of the AD control software.

Cybersecurity Value: Vehicle dynamics data enables a greater understanding of the effect of cyber attacks on vehicle behavior. The utility of vehicle dynamics data includes research and development of physical intrusion detection system solutions and root cause analysis.

Data Availability: We are not aware of any datasets for vehicle dynamics in the context of cybersecurity.

B. Gaps in Autonomous Vehicle Datasets

Our exploration of diverse AD data types and their usage in cybersecurity has identified a number of limitations:

- **Lack of a consolidated research data platform.** Datasets are distributed across GitHub accounts and research papers. There is a lack of consolidation of datasets that would enable security research across the AD technology stack.
- **Siloed research.** Defensive mechanisms are often developed based on a single data type (e.g., CAN, Camera, etc.). The lack of availability of other data sources and an understanding of how this data impacts vehicle dynamics and propagates through the AD system results in the

creation of defense mechanisms that lack system-level validation.

- **Lack of cybersecurity data:** There is a lack of data for cybersecurity, and in some of the sub-categories explored, there is, to our knowledge, no data available. The available datasets overwhelmingly consist of legacy and connected vehicles.

III. ADSecData Platform

In developing a method for generating cybersecurity data for AD systems, the significant change from legacy vehicles is the focus on vehicle behavior. As the vehicle is controlled by software and algorithms, it is important to understand the effect of cyber activity on the vehicle and its implications for decision control. In addition to attacks that directly target AD technologies, such as advanced sensors, attacks on network and system components can have a downstream effect on autonomous control. The ADSecData Platform (shown in Fig. 1) follows a four-stage process for generating data.

A. *Scenario Generation*

Scenario-based testing (SBT) involves evaluating the performance of a module or the full AD pipeline (perception, localization, planning, and decision-control) to perform its task during a specified driving scenario. Since the performance of algorithms can vary under diverse scenarios, SBT has become the standardized approach for AD algorithm safety validation and verification testing [32]. Cybersecurity represents an edge and corner case for SBT. For the ADSecData methodology, we propose that scenario generation is a crucial step for cybersecurity, as it is essential to understand whether the effect of a cyber attack on the vehicle differs based on the scenario. Since scenario libraries for AD cybersecurity testing are not available, our methodology recommends using safety validation testing libraries (such as ASAM OpenScenario, etc.) and customizing the scenarios with attack models.

B. *Simulation/Test Environment*

As the task of driving can encounter many diverse scenarios, simulation is the only feasible mechanism to incorporate large-scale testing agilely. Cybersecurity testing should be aligned with safety validation testing, where the choice of test environment is based on evaluating the algorithm's ability to perform tasks. This is part of a testing process that uses regression testing to map scenario test sets from simulation test environments to real-world proving grounds. Within the ADSecData platform, we recommend using low-fidelity test environments for large-scale testing of driving logic, high-fidelity test environments to include testing of advanced sensors (such as LiDAR, Camera, etc.), and real-world proving grounds. Another factor influencing the integrity of cybersecurity data is the tendency of automotive cybersecurity practitioners to provide singular datasets based on attack type. Due to the experimental nature of AD algorithms, sufficient tests need to be run to ensure that anomalous vehicle behavior is caused by cyber activity and not system errors or a lack of algorithm optimization.

Another key aspect of the simulation/test environment stage is defining metrics and configuring the format of output data. Safety metrics and vehicle dynamic parameters are applied

TABLE I: Requirements for ADSecData

Category	**Requirement**
Documentation	• Dataset should be accompanied by general documentation describing content and origin. • Documentation should include description of the attacks in the dataset and how they were executed/recorded. • Documentation should include description of the features (e.g., origin, meaning, range) and their physical context (e.g., how vehicle speed, engine speed and gear are related).
Labels	• Each entry in the dataset may be given a label for identifying whether that entry is benign or an attack.
Parseability, correctness and consistency	• Data should be stored in an appropriate machine/humanreadable format (e.g., PCAP or CSV rather than SQL databases) • All entries should be correctly formatted (e.g., no corrupt entries) • use a single data format for all entries
Age, Size, Objective	• Dataset should not be legacy ($>$ 5 years old etc.) and consist of a balance between benign and cyber attack data.
Completeness	• Dataset should be complete in the sense that no key features or entries have been discarded.
Transformation and anonymization	• Data should not be irreversibly transformed (changing timestamps etc.) and not be anonymised to the point that it bias' detection mechanisms.
Dataset and Attack Realism	• Dataset should include diverse attacks and not be wholly based on synthetic data.

to quantify the impact of cyber activity on the vehicle. Cybersecurity labels include details such as the attack initiation during the scenario, attack parameters (e.g., sensor interference noise level, GPS positioning offset), and their corresponding weighting.

C. *Analysis*

The analysis stage involves interrogating the data to assess its integrity and accuracy, ensuring consistency with the experimentation performed. Popular tools, including MATLAB and Python, are used to plot data, visualize patterns, and analyze trends. For example, analyzing a dataset from the trajectory planning module could generate trajectory maps to visualize the vehicle's path and highlight any deviations from the reference path. Analysis is a crucial activity for identifying problems with the experimentation process and evaluating the data quality.

D. *ADSecData*

Data should be benchmarked for measurement and comparison. The benchmarks for automotive cybersecurity datasets from Vahidi et al. [33] systematic evaluation of automotive intrusion datasets serve as a good starting point. We utilize their data requirements to develop the ADSecData Platform and data readiness labels. Table. I provide the requirements for ADSecData datasets, the classification is provided in more detail on the platform website.

IV. ADSecData Case Study

A. *Target Autonomous Vehicle*

The target vehicle is an AV for public transportation, that is an autonomous electric vehicle (AEV). The shuttle operates at Level 4 autonomy (high automation), meaning that it can handle most driving tasks without human intervention in predefined areas, and it is equipped with advanced LiDAR, radar, cameras, and GPS systems to navigate safely and carry out perception tasks in an urban environment. Its software

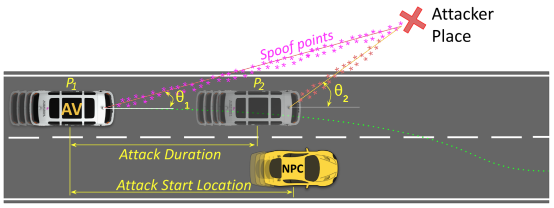


Fig. 2: Attack Case 1 Threat Model.

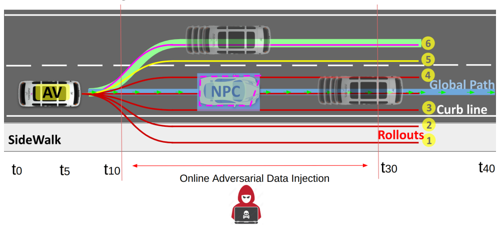


Fig. 3: Attack Case 2 & 3 Threat Model.

backbone is based on ROS and Autoware, controlling all the driving functionalities and implementing the driving dynamic model of the vehicle.

B. Scenarios

Our initial dataset consists of 4 attack cases conducted during diverse driving scenarios.

Attack Case 1 - LiDAR point-cloud manipulation: The LiDAR point-cloud manipulation attack, as shown in Fig. 2, consists of an adversary with a LiDAR capable of injecting malicious LiDAR point clouds into the LiDARs of the AV. This attack is conducted whilst the AV is attempting an overtaking maneuver.

Attack Case 2 - Position Offset: Attack Case 3 - Message Delay: The attacker creates a spoofed ROS topic which is able to deliver malicious input data of the `Current_Pose` (longitude, latitude, and velocity) to all the nodes of the local planning module. The data manipulation is injected online/dynamically during the critical overtaking manoeuvre involving the AV and NPC (Non-playable character). Figure 3 displays the critical driving scenario and the time frames in which the manipulated `Current_Pose` data is injected into the local planning pipeline cost estimation. The red dashed lines in Fig. 3 represent the roll-outs, and the green highlighted, denoting the selected motion-path.

For the manipulation of the `Current_Pose` data, we introduce a deviation to lateral and longitudinal pose. For the lateral pose data, the sensitivity deviation introduced was structured as follows:

- Attack Case 2a: 0.16%
- Attack Case 2b: 0.33%
- Attack Case 2c: 0.5%

This range represents a slight perturbation of pose to a 1m deviation. The longitudinal pose data sensitivity deviation range was structured as follows:

- Attack Case 2d: 0.33%
- Attack Case 2e: 0.66%
- Attack Case 2f: 1.00%

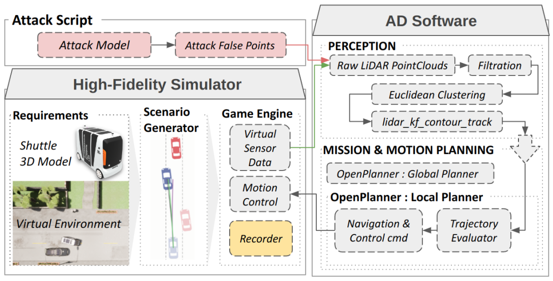


Fig. 4: Architecture of the testing platform.

This range is the same as the longitudinal deviation. The difference in percentage comes from the difference in coordinate values of lateral and longitude. The lateral value is almost double that of the longitudinal, and therefore, the percentage is doubled.

This attack scenario involves introducing a time-delay into the messages of the Current_Pose topic communicating to the nodes of the local planning module.

We introduced a message delay when the AV passes 2m in front of the vehicle that it is passing in the lateral direction. We introduce 3 different time delays in the message:

- Attack Case 3a: 0.3 seconds
- Attack Case 3b: 0.6 seconds
- Attack Case 3c: 1.0 seconds

The message frequency is approximately 50hz, so this is a message every 20 milliseconds. We chose the above range of deviation of time-delay as it enabled a spectrum of a message from the delay from approximately 15, to 50 messages.

Attack Case 4 - GPS Spoofing: The attack model of GPS spoofing involves an adversary using a transmitter near the AV and interferes with the GPS signals being transmitted.

C. Simulation/Test Environment

Attack Case 1 was conducted in the high-fidelity CARLA simulator [34]. In this study, we use Carla 0.9.13 as the high-fidelity simulator. Figure 4 illustrates the requirements for the high-fidelity simulator to conduct simulation testing, which are two components, the digital twin of the target AV and the virtual replication of our target environment. These replicated components help us to gain more accurate results of the proposed platform [35]. The AV digital twin is a 3D model of the target real-world world AV shuttle, designed in Blender, a graphical 3d modeling software, and imported and built in Unreal for deployment in CARLA. This model uses the same dimension and sensor configuration (model, position, and orientation) from the real AV shuttle. The environment digital twin, in our case, is identical to the location where the vehicle operates.

This simulation setup was implemented on a desktop computer with the following configuration:

- Intel® Core™ i7-11700K @ 3.60GHz × 16 cores
- NVIDIA GeForce RTX 3080 10 GB
- RAM: 128 GB

Attack Case 2 and 3 were conducted in a low-fidelity simulator. To accelerate the testing, we bypassed the sensing and detection nodes of the algorithm and focused on the planning part by utilizing the low-fidelity simulation feature provided by Autoware.ai and Openplanner. The low-fidelity simulation uses the open-planner 2.5 control algorithm. It provides simulated localization and detection data for the planning nodes and receives the actuation commands to simulate the AV kinematics. This process runs faster due to the low-detail environment required for the simulation and the lack of the process to simulate the sensors.

Attack Case 4 dataset was generated from the real-world vehicle. GPS spoofing activity occurred during a point-in-time of a 3 month trial of AVs in a city in Northern Europe.

D. Analysis

The data output parameters were defined based on safety, vehicle dynamics and security criteria. A sample of these includes safety criteria, mission success, violation, break status, and distance-to-collision. Vehicle dynamics included steer, yaw, lateral and longitudinal position. Security criteria include 2 labels, *is_attack* denoting when the attack is occurring and *cyber_weight*, which denotes the level of sensor noise manipulation.

E. ADSecData

The 4 attack case scenarios datasets were generated as a .csv files. Each attack includes a corresponding benign (no attack) dataset to benchmark the stability of the AD algorithms under the given driving scenario. *Attack Case 1* included over 1200 simulations. *Attack Case 2 and 3* included over 900 simulations collectively. The data is available at this link: ADSecData Platform (*To note: this website is abridged, anonymised version*)

V. Discussion

The case study provides a starting point for the development of a common dataset for the community to perform fair and reproducible evaluations of AD algorithms for cybersecurity and defensive mechanisms. The datasets generated from the 4 attack cases demonstrate the importance of following the 4 stage ADSecData method where particular careful consideration is taken in the definition of data output parameters and experimental evaluation analysis. For the development of the ADSecData platform, community challenges, and a roadmap are fundamental.

A. Community Challenges

These are the first tranche of community challenges that we recommend for the ADSecData platform:

Ch1 Performance and Accuracy of Semantic Fuzzing Tools.
Ch2 Intrusion Detection of Semantic AD Sensor Attacks.
Ch3 Robust Sensor Fusion Algorithms.
Ch4 Robust and Resilient Trajectory Planning Algorithm.

We see these challenges as of most immediate importance and value for the community. Furthermore, we would like to see the community use the ADSecData platform to generate a seed corpus for guided semantic data fuzzing tools. As large language models (LLMs) are gaining in popularity, another foreseeable use would be to apply LLMs to ADSecData to generate scenarios for cybersecurity testing. As AD cybersecurity lacks a common scenario library, the generation of cybersecurity scenarios would help to close this gap. Finally, IDS solutions for attacks on the AD sensors are essential to mitigate the risk to AD control. There needs to be more data to understand the profile of cyber attacks compared to emergency and safety actions from edge and corner cases.

B. Future Roadmap

The short-term aims of the ADSecData platform are to add more datasets from all 4 sub-categories of data types and different vehicle classes and increase the community's awareness of the platform. There will be a need to improve the development of both the front-end and back-end platforms to enable secure data sharing and a more intuitive user experience. Longer-term aims include a need to investigate metrics for intrusion detection solutions for AD, which is an AI-based system. Traditionally, MITRE ATT&CK is used for benchmarking IDS solutions, and MITRE has a framework for AI, MITRE ATLAS. It would be interesting to evaluate how this would work in a practical use-case for AD.

VI. Related Work

There have been attempts by the community to build common infrastructure for AV cybersecurity testing. PASS [36] and Simutack [37] are community simulation testing platforms. Whilst these platforms are valuable to the community and enable accessibility of simulation testing to researchers, the usage of community simulation testing platforms is limited as real-world operators tend to use their own customised platforms. Furthermore, neither of these studies focused on the data aspect of cybersecurity testing as part of their scope. Lauinger et al. [38] developed an attack data generation framework for AVs. Our work enhances this contribution by integrating the concepts of scenario generation and simulation and testing environments for data generation.

From a community data sharing perspective, there are initiatives such as Platform for Innovative use of Vehicle Open Telematics (PIVOT) [39], which is a U.S National Science Foundation project to create a open-source portal for vehicle telemetry data in the context of cybersecurity. However, as of writing this portal was unavailable.

As aforementioned in Section. II, there exists a diversity of datasets for legacy and connected vehicles. There are also the studies of Vahidi et al. [33], Lampe & Meng [25] and Lee et al. [40] which evaluate cybersecurity data of legacy and connected vehicles for intrusion detection. However, to our knowledge, there are no existing contributions that focus on the autonomous technology stack of AVs.

VII. Conclusion

In this work, we present an open-source data platform for AD cybersecurity, the ADSecData platform. Further, we detail a 4-stage method for the generation of AD cybersecurity datasets. We demonstrate the utility of the ADSecData platform through the generation of open-source datasets for four diverse AD cybersecurity attacks, which we provide to the community. The ADSecData platform is available to the community and will continue to develop according to the challenges and roadmap presented in this study.

Acknowledgment

This work is partly supported by the Horizon Europe projects CyberSecDome (Agreement No.: 101120779) & XTRUST-6G (Agreement No.: 101192749) and the European Union and Estonian Research Council via project TEM-TA5.

References

[1] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[2] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[3] P. Wang, X. Huang, X. Cheng, D. Zhou, Q. Geng, and R. Yang, "The apolloscape open dataset for autonomous driving and its application," *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[4] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. K. Pontes, D. Ramanan, P. Carr, and J. Hays, "Argoverse 2: Next generation datasets for self-driving perception and forecasting," in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks 2021)*, 2021.

[5] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *CVPR*, 2020.

[6] CARLA, "Carla autonomous driving leaderboard," 2024. [Online]. Available: https://leaderboard.carla.org/

[7] W. Xu, H. Zhou, N. Cheng, F. Lyu, W. Shi, J. Chen, and X. Shen, "Internet of vehicles in big data era," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 1, pp. 19–35, 2018.

[8] Y. Cao, C. Xiao, B. Cyr, Y. Zhou, W. Park, S. Rampazzi, Q. A. Chen, K. Fu, and Z. M. Mao, "Adversarial sensor attack on lidar-based perception in autonomous driving," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19. Association for Computing Machinery, 2019.

[9] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1625–1634.

[10] T. Sato, S. H. Bhupathiraju, M. Clifford, T. Sugawara, Q. A. Chen, and S. Rampazzi, "Wip: Infrared laser reflection attack against traffic sign recognition systems," *ISOC Symposium on Vehicle Security and Privacy (VehicleSec)*, 2023. [Online]. Available: https://par.nsf.gov/biblio/10427118

[11] R. S. Hallyburton, Y. Liu, Y. Cao, Z. M. Mao, and M. Pajic, "Security analysis of Camera-LiDAR fusion against Black-Box attacks on autonomous vehicles," in *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 1903–1920. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/hallyburton

[12] R. Jiao, J. Bai, X. Liu, T. Sato, X. Yuan, Q. A. Chen, and Q. Zhu, "Learning representation for anomaly detection of vehicle trajectories," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 9699–9706.

[13] Y. Jia, Y. Lu, J. Shen, Q. A. Chen, H. Chen, Z. Zhong, and T. Wei, "Fooling detection alone is not enough: Adversarial attack against multiple object tracking," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=rJl31TNYPr

[14] R. Quinonez, J. Giraldo, L. Salazar, E. Bauman, A. Cardenas, and Z. Lin, "Savior: securing autonomous vehicles with robust physical invariants," in *Proceedings of the 29th USENIX Conference on Security Symposium*, ser. SEC'20. USA: USENIX Association, 2020.

[15] T. Sato, J. Yue, N. Chen, N. Wang, and Q. A. Chen, "Intriguing Properties of Diffusion Models: An Empirical Study of the Natural Attack Capability in Text-to-Image Generative Models," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[16] C. Ma, N. Wang, Q. A. Chen, and C. Shen, "SlowTrack: Increasing the Latency of Camera-Based Perception in Autonomous Driving Using Adversarial Examples," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 5, 2024, pp. 4062–4070.

[17] I. M. G. Costantino, M. De Vincenzi, "A vehicle firmware security vulnerability: an ivi exploitation." *J Comput Virol Hack Tech*, vol. 20, pp. 681,696, 2024.

[18] Turla, "Mazda getinfo attack," 2017. [Online]. Available: https://github.com/shipcod3/mazda_getInfo

[19] M. D. Pesé, T. Stacer, C. A. Campos, E. Newberry, D. Chen, and K. G. Shin, "Librecan: Automated can message translator," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 2283–2300. [Online]. Available: https://doi.org/10.1145/3319535.3363190

[20] N. Alkhatib, M. Mushtaq, H. Ghauch, and J.-L. Danger, "Can-bert do it? controller area network intrusion detection system based on bert language model," in *2022 IEEE/ACS 19th International Conference on Computer Systems and Applications (AICCSA)*, 2022, pp. 1–8.

[21] E. Seo, H. M. Song, and H. K. Kim, "Gids: Gan based intrusion detection system for in-vehicle network," in *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, Aug 2018, pp. 1–6.

[22] H. M. Song, J. Woo, and H. K. Kim, "In-vehicle network intrusion detection using deep convolutional neural network," *Vehicular Communications*, vol. 21, p. 100198, 2020.

[23] M. L. Han, B. I. Kwak, and H. K. Kim, "Anomaly intrusion detection method for vehicular networks based on survival analysis," *Vehicular Communications*, vol. 14, pp. 52–63, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214209618301189

[24] B. Lampe and W. Meng, "can-train-and-test: A curated can dataset for automotive intrusion detection," *Computers & Security*, vol. 140, p. 103777, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404824000786

[25] ——, "can-train-and-test: A new can intrusion detection dataset," in *2023 IEEE 98th Vehicular Technology Conference (VTC2023-Fall)*, 2023, pp. 1–7.

[26] M. Hanselmann, T. Strauss, K. Dormann, and H. Ulmer, "Canet: An unsupervised intrusion detection system for high dimensional can bus data," *IEEE Access*, vol. 8, pp. 58 194–58 205, 2020.

[27] A. Gazdag, R. Ferenc, and L. Buttyán, "Crysys dataset of can traffic logs containing fabrication and masquerade attacks," *Scientific Data*, vol. 10, 2023.

[28] E. C. P. Neto, H. Taslimasa, S. Dadkhah, S. Iqbal, P. Xiong, T. Rahman, and A. A. Ghorbani, "Ciciov2024: Advancing realistic ids approaches against dos and spoofing attack in iov can bus," *Internet of Things*, vol. 26, p. 101209, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660524001501

[29] S. Rajapaksha, G. Madzudzo, H. Kalutarage, A. Petrovski, and M. O. Al-Kadri, "Can-mirgu: A comprehensive can bus attack dataset from moving vehicles for intrusion detection system evaluation," in *Symposium on Vehicles Security and Privacy. Internet Society*, 2024.

[30] S. Iqbal, P. Ball, M. H. Kamarudin, and A. Bradley, "Simulating malicious attacks on vanets for connected and autonomous vehicle cybersecurity: A machine learning dataset," in *2022 13th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, 2022, pp. 332–337.

[31] F. Gonçalves, B. Ribeiro, O. Gama, J. Santos, A. Costa, B. Dias, M. J. Nicolau, J. Macedo, and A. Santos, "Synthesizing datasets with security threats for vehicular ad-hoc networks," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–6.

[32] Y. Huai, Y. Chen, S. Almanee, T. Ngo, X. Liao, Z. Wan, Q. A. Chen, and J. Garcia, "Doppelgänger test generation for revealing bugs in autonomous driving software," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, 2023, pp. 2591–2603.

[33] A. Vahidi, T. Rosenstatter, and N. I. Mowla, "Systematic evaluation of automotive intrusion detection datasets," in *Proceedings of the 6th ACM Computer Science in Cars Symposium*, ser. CSCS '22. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: https://doi.org/10.1145/3568160.3570226

[34] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[35] M. Malayjerdi, V. Kuts, R. Sell, T. Otto, and B. C. Baykara, "Virtual simulations environment development for autonomous vehicles interaction," in *ASME International Mechanical Engineering Congress and Exposition*. American Society of Mechanical Engineers, 2020.

[36] Z. Hu, J. Shen, S. Guo, X. Zhang, Z. Zhong, Q. A. Chen, and K. Li, "Pass: A system-driven evaluation platform for autonomous driving safety and security," *NDSS Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*, 2025. [Online]. Available: https://par.nsf.gov/biblio/10359464

[37] A. Finkenzeller, A. Mathur, J. Lauinger, M. Hamad, and S. Steinhorst, "Simutack - an attack simulation framework for connected and autonomous vehicles," in *2023 IEEE 97th Vehicular Technology Conference (VTC2023-Spring)*, 2023, pp. 1–7.

[38] J. Lauinger, A. Finkenzeller, H. Lautebach, M. Hamad, and S. Steinhorst, "Attack data generation framework for autonomous vehicle sensors," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2022, pp. 128–131.

[39] P. Project, "Platform for innovative use of vehicle open telematics," 2024. [Online]. Available: https://pivot-auto.org/

[40] S. Lee, W. Choi, I. Kim, G. Lee, and D. H. Lee, "A comprehensive analysis of datasets for automotive intrusion detection systems," *Computers, Materials and Continua*, vol. 76, no. 3, pp. 3413–3442, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1546221823000516

Appendix III

Paper III

A. Roberts, J. Cheng, O. Maennel, M. Hamad, and S. Steinhorst. Adseclang: A domain-specific language for cybersecurity testing of autonomous vehicles. In 2025 IEEE 101st Vehicular Technology Conference (VTC2025-Spring), pages 1–6, 2025.

ADSecLang: A Domain-Specific Language for Cybersecurity Testing of Autonomous Vehicles

Andrew Roberts[1], Jingyue Cheng[2], Olaf Maennel[3], Mohammad Hamad[2], Sebastian Steinhorst[2]
[1]FinEst Centre for Smart Cities, Estonia
[2]Technical University of Munich,Germany
[3]University of Adelaide, Australia

***Abstract*—Domain-specific languages for safety validation testing have reduced the complexity of safety scenario generation and enhanced the adoption of Autonomous Driving (AD) safety testing. Yet, there is a lack of comparable solutions for cybersecurity testing. In this work, we present *ADSecLang*, a domain-specific language for cybersecurity testing of AD systems. *ADSecLang* provides a concise syntax that enables the tester to construct scenarios for AD cybersecurity straightforwardly that can be implemented in an AD test simulation platform. The proposed language is validated within the CARLA AD simulation platform in a use-case scenario of two diverse AV camera sensor manipulation attacks on a state-of-the-art trajectory-guided AD solution. The results show that the language is able to support the translation of threats from an abstract description to the technical implementation of the attack test cases and that these test cases could identify vulnerabilities in the target AD solution.**

***Index Terms*—Autonomous Driving, Cybersecurity testing**

I. Introduction

Vulnerability testing of autonomous driving (AD) to cyber attacks is a burgeoning field of research. Initial contributions to this field have focused on novel vulnerability discovery utilising penetration testing methods [1] [2] and fuzzing [3], [4]. However, there exists a gap between this novel, experimental work and the practical implementation of testing to validate the operational readiness of real-world AD systems. Real-world, operational AV testing requires a more rigid approach centred on a structured testing methodology aligned to composite vehicle development and test validation processes. For safety validation testing, domain-specific languages for safety scenario generation, such as SCENIC [5] and ASAM OpenSCENARIO [6], provide a systematic expression that enables a common taxonomy, traceability of testing processes and repeatability and automation of testing for scalability. Yet, there exists a sparsity of research on the development of a domain-specific language for cybersecurity testing of AD systems. One of the primary benefits of the development of a domain-specific language for cybersecurity is that it can simplify the task of writing scenarios for security by providing a concise syntax. In addition, the lack of a domain-specific taxonomy for cybersecurity testing of AD systems further challenges the development and evaluation of AD security testing tools, processes, and methods.

This paper aims to develop such a language, which we call ADSecLang. ADSecLang acts as an intermediary layer in the testing process, which constructs scenarios for cybersecurity through the translation of functional threat descriptions to concrete test cases. Figure 1 depicts the scenario-based abstraction of ADSecLang, which represents the incremental and iterative definition of the threat scenario. Firstly, the abstract description of the threat scenario originates from adversarial analysis, which can leverage sources such as threat libraries, system data, and other knowledge-base repositories. Secondly, a logical, syntactical expression of the threat scenario is created using a taxonomy. Finally, the technical description of the threat scenario is integrated within the AD simulation testing platform. ADSecLang aims to contribute greater intuition through readable, concise syntax for the development of adversarial agents in simulation testing that would otherwise require complex expressions and constraints. ADSecLang requires the tester to consider all elements of the threat model from attacker tools to desired attack outcomes at both an abstract and parameterised level of abstraction. To demonstrate the utility of ADSecLang, we initially focus on semantic AI security and we evaluate the language to support two use-case scenarios of a camera manipulation attack. In summary, the main contributions of our work are:

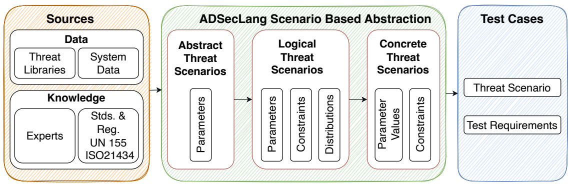


Fig. 1: ADSecLang: scenario-based abstraction view.

- We provide an *Attack Taxonomy* which provides common knowledge for the construction of cybersecurity test scenarios (Sec. II-A).
- We present *ADSecLang*, a domain-specific language for autonomous vehicle cybersecurity testing (Sec. II).
- We use *ADSecLang* to identify vulnerabilities in a state-of-the-art trajectory-guided end-to-end AD solution (Sec. III).

II. ADSecLang: The Proposed Solution

This section introduces the attack taxonomy used to support the development of ADSecLang and presents the cybersecurity testing framework where ADSecLang can be adopted.

A. Attack Taxonomy

The attack taxonomy of ADSecLang (as shown in Fig. 2) categorizes cyber attacks into two domains: **Action** and **Impact**.

1) Action: represents the execution of an attack method. The success of an action depends on the fulfillment of one or more preconditions. As a result, we subdivide the Action domain into two sub-domains: **Method** and **Preconditions**. The **Method**

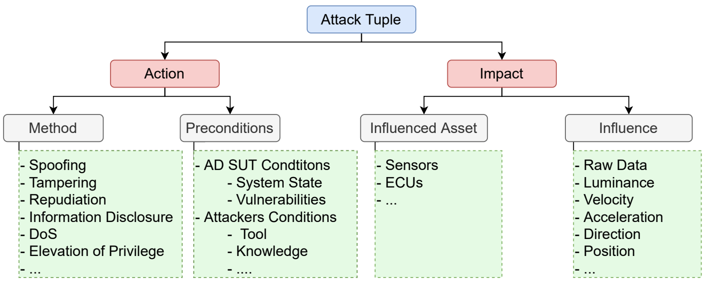


Fig. 2: Attack Taxonomy - Detailed Description.

is defined as the threat technique. This threat description can be derived from a functional description such as STRIDE, Attack Trees, or a textual interpretation. **Preconditions** are a set of conditions that must be met to execute an attack. These preconditions must be inherent attributes that already exist and are not generated by the execution of the attack. The preconditions can be further divided into two categories: conditions on the AD System-Under-Test (SUT) and conditions on the attacker.

- *AD SUT Conditions* are categorized into requirements for the state of the tested system and vulnerabilities within the system. System state conditions refer to the requirement that the target system must be in a specific state (such as a particular version of an operating system, system software/application, or a specific hardware/software state, such as firmware update status) for the attack to be executed. System vulnerabilities refer to exploitable weaknesses in the system's design and operation.
- *Attacker Conditions* can be further segmented into three types: attack tools, attacker knowledge (capabilities, skills), and the level of privileges that an attacker can obtain. The tools and knowledge of the attacker help to profile the type of threat actor capable of conducting the attack. The level of privileges refers to the permissions needed to access or manipulate target system assets. An example would be permission to run processes on the target or existing access to the target asset to manipulate data.

Some attack methods can only be executed successfully when multiple preconditions are met simultaneously. Such conditions will be grouped within braces {}. For example, the precondition `[{A, B}, C]` can be interpreted as 'A and B must be met simultaneously, or C must be met'. To encompass the requirement for multiple preconditions, we define an **Action Group**:

```
action: [method, preconditions]
   method: [category, description]
   preconditions: [precond1, precond2, ...]
     precond1: [category, description]
     precond2: [category, description]
```

2) Impact: Executing an *Action* will introduce one or more **Impacts** into the system. In other studies [7], these impacts are also denoted as consequence and effect. Although *Impacts* represent the outcomes and effects of attacks, they can also serve as preconditions for subsequent attacks. Consequently, some researchers [8] have alternatively referred to them as post-conditions. In our taxonomy, the utilization of *Impact* aims to identify the direct consequences of an *Action*, which may additionally serve as preconditions for further attacks. The term 'goal' in the attack model represents the ultimate impact. The dimension of *Impact* can be subdivided into two sub-dimensions, namely **Influenced Assets** and **Influence**, which serve to identify the assets directly affected by the *Action* and ascertain the direct impact incurred on these assets. **Influenced Assets** can be characterized by their respective category and name. For example, the sensor category can include cameras, radars, LiDAR, GPS, or any other AD sensor. The electronic control unit (ECU) category comprises brake control ECUs, engine control ECUs, tire pressure monitoring ECUs, or any other vehicular ECU. **Influence** can be specified as its *Parameter* and *Value*, denoting the specific parameter influenced by the attack and the corresponding altered value, respectively. For instance, if we aim to adjust the brightness of an image captured by a camera, we should specify the parameter as luminance and set its value to 300% (indicating that the brightness has been increased to 300% of the original brightness). To achieve the scalability of ADSecLang, users can add new parameters and a value range in the property configuration file. The **Impact Group** is defined as follows:

```
impact: [influenced_asset, influence]
   influenced_asset: [category, name]
   influence: [parameter, value]
```

B. Semantics of ADSecLang

The safety scenario domain-specific languages are based on scenario abstraction methodologies such as the Pegasus method [6], which segments three levels of abstraction of the scenario: 1) abstract, 2) logistic, and 3) concrete. For example, an abstract scenario could be described as: '*A malicious actor motivated to cause a safety violation using a laser beam device targeted at a car*'. The logical scenario might be: '*A malicious actor using a laser beam device projecting a luminance of approximately 100 to 300% with a pulse width of 0 to 1*'. Finally, the concrete scenario would specify: '*A malicious actor using a laser beam device projecting 300% luminance with a pulse width of 1*'. Within ADSecLang, the abstract describes the cyber threat scenario according to local parameters. The logical cyber threat scenario extends this description by adding parameter value ranges. Finally, the concrete scenario description contains the set parameter values, which will be utilized as the scenario implementation within the AD simulation testing platform. ADSecLang is designed as an extension of the safety scenario

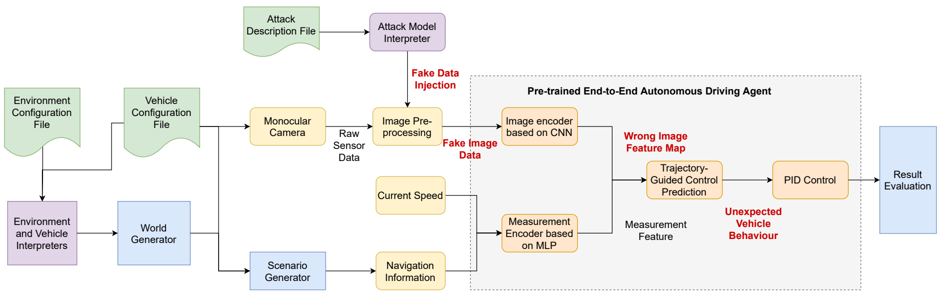


Fig. 3: ADSecLang Cybersecurity Testing Workflow - Camera Attack.

languages [5], [6], using the same abstraction method, language semantics and syntax. ADSecLang provides an extension to these areas for cybersecurity.

C. Compilation of ADSecLang

Compilation of ADSecLang involves three configuration files. Each file contains various user-defined parameters:

- *Environment Configuration File*: This file provides adjustable parameters for scene generation, including town, weather, and traffic density. It also allows users to define constraints on these parameters for scene sampling.
- *Vehicle Configuration File*: This file allows the user to define the parameters of the autonomous vehicle; these include the sensors required, the location of the sensors in the vehicle, the type of sensors, the data to be recorded, and the frequency of recording.
- *Attack Description File*: This file is formatted in the YAML syntax and allows users to define the attack model.

The first two configuration files are relatively simple: the *Environment Configuration File* and the *Vehicle Configuration File*. The environment and vehicle configurations stored in their respective configuration files are read as parameters for generating the driving simulation world and transferred to the world generator. The *Attack Description File* is a more complicated design which has two functions:

- The attack description file is utilized to extract the parameters, which are then translated into concrete code implementation for data processing based on the corresponding attack parameters.
- It is also responsible for connecting the simulation environment, attack code, and autonomous driving system. The attack description file defines the input and output interfaces of the attack code. The input interface connects real-time data captured by sensors in the rendering engine in a simulation environment, such as images captured by camera sensors or status information of ECUs. The output interface sends malicious data generated by attacks to the target AD solution.

D. Cybersecurity Testing Framework

1) Architecture: The proposed cybersecurity testing framework has diverse modules for environmental, vehicle, and attack configuration, simulation test, and result evaluation (Fig. 3). The functions and roles of these modules are as follows:

- *Environment and Vehicle Interpreter:* Reads the environment and vehicle configuration stored in their respective configuration files as a parameter for generating the world.
- *Attack Model Interpreter:* We read the attack description file as attack parameters. We have defined input and output interfaces for the attack model. The input interface obtains images captured by sensors in the real-time rendering engine and completes the data processing corresponding to the attack parameters read by the interpreter in the specific implementation code of the interface. The specific implementation of the output interface is to send the output of the attack model to the user's chosen autonomous driving solution.
- *World Generator:* Initialise the world based on the environment and entity parameters read by the environment interpreter, including object properties and attribute distribution functions. The world generator randomly samples from the distribution function whenever it is called. By reading the sampling results of the world initializer, a specific world is generated in the real-time rendering engine according to certain steps. The generated world contains at least one vehicle and one camera sensor and exposes the calling interfaces of the vehicle and sensors to the attack model.
- *Scenario generation and result evaluation:* We use a CARLA plugin called CARLA Leaderboard [9] to provide us with scenario generation and evaluation of driving violations. Violation testing includes route completion testing, collision testing, red light running testing, stop running testing, lane crossing testing, proxy blocking testing, and timeout testing.

2) Cybersecurity Testing Workflow: The overall workflow of the system is shown in Fig. 3. The attack target system illustrated here is an end-to-end autonomous driving system based on a monocular camera. The target asset in the vehicle of the attack is the monocular front RGB camera.

The workflow is initiated by storing the predefined environment configuration, object properties, and attack description in configuration files. Execution of the *World Generator* uses the *Environment and Vehicle Interpreter* to read the environment information. Subsequently, each time the scenario is generated, sampling is carried out according to the predetermined process, and the sampling results are converted into the parameterized form we designed and then handed over to the *World Generator*. The *World Generator* first initializes the basic configuration of the real-time rendering engine and creates a specific world in the it, step by step, based on the obtained parameters. Once the world is created, the *Scenario Generator* starts generating test scenarios based on the preset parameters. Subsequently, the *Attack Model Interpreter* retrieves the attack information from the *Attack Description File* and injects the manipulated data to the end-to-end AD system based on the parameters specified by the attack model. Finally, the *Results Evaluation* checks conformity of the AV to safety metrics, as aforementioned, as part of the CARLA Leaderboard [9]. Through conducting multiple iterations of the testing workflow it is possible to evaluate the effectiveness of the attack model.

III. Evaluation Case Studies

This section examines the use of ADSecLang for supporting the security testing of AV systems. It includes a description of the experimental setup (Sec. III-A) and an analysis of results derived from two attack scenarios (Sec. III-B and III-C). The goal of the experiments is to assess the ability of ADSecLang to generate attack test cases capable of identifying vulnerabilities in AD systems.

A. *Experimental Setup*

The experiments were run on a desktop computer with 12th Gen Intel(R) Core(TM) i3-12100F 4-Core Processor, NVIDIA GeForce GTX 1070Ti GPU, and 16 GB RAM. The use-case scenario testing is conducted on the simulator CARLA 0.9.10. The AD solution tested in the following experiments is a trajectory-guided end-to-end AD solution [10]. This AD solution achieves a new state-of-the-art performance on the CARLA AD Leaderboard [9], in which they rank first in terms of the Driving Score and Infraction Penalty using only a single camera as input. The image captured by the camera has a resolution of 900×256 pixels, and the field of view is maximized at 100 degree.

B. *Attack Case I - Strong Light Exposure Attack*

1) Attack Design: State-of-the-art camera attacks [11] have shown that strong white LED light directed at the camera sensor will result in significantly higher hue values and cause the entire image to be completely white. This results in the camera being unable to capture any visual information. This attack is based on the fact that CMOS/CCD sensors can be interfered with by malicious optical inputs and will produce unrecognizable images. The broken image will further affect the victim AV's decision control. As a result, it will cause uncertainties, which may lead the victim's AV to deviate or emergency brake, both of which can lead to a collision and/or other safety violations. Common methods of attacking camera devices are lasers or LEDs. The Strong Light Exposure Attack interferes with the camera's automatic exposure control. Under laser irradiation, the surface temperature will rise rapidly due to the non-uniform temperature field. Avalanche breakdown of semiconductor materials will cause irreversible damage to optoelectronic devices. Whilst we cannot achieve the physical effects of a targeted light on the camera sensor in a virtual simulator, we can modify the data to simulate the profile of the cyber-physical attack.

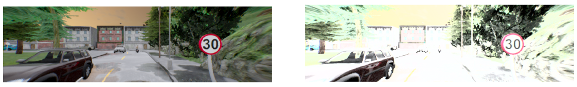

(a) Before (b) After

Fig. 4: Camera view of attack case 1: before (a) and after (b) the implementation of the Strong Light Exposure Attack.

2) ADSecLang Attack Configuration: The concrete scenario using the ADSecLang attack interpreter file is provided below.

```
attack_name: strong light exposure attack
attack_target: monocular camera-based end-to-end autonomous driving system
attack_goal: safety hazard
    action: [method, preconditions]
         method: [tampering, modifying the data captured in the asset]
         preconditions: [{precond1 AND precond2 AND precond3}]
          precond1: [attacker's knowledge, the attacker knows the basic information about the cameras on the victim's autonomous driving vehicle]
          precond2: [attack tool, strong LED light]
          precond3: [attacker's capability, attackers can shine LED light at AV camera sensor]
    impact: [influenced_asset, influence]
         influenced_asset: [sensor, rgb_camera_front]
         influence: [luminance, 300%]
```

The attack description YAML file is translated using the attack interpreter within the simulation platform.

```
    if(config['attack name']=="Strong light exposure attack"):
    percentage = config['impact']['influence']['luminance']
    file.write('    data = cv2.cvtColor(data, cv2.COLOR_RGB2YUV)\n')
    file.write('    h = data.shape[0]\n')
    file.write('    w = data.shape[1]\n')
    file.write('    for i in range(h):\n')
    file.write('        for j in range(w):\n')
    file.write('            y = data[i][j][0]*'+str(float(percentage[:-1]) / 100.0)+'\n')
    file.write('            if y > 255:\n')
    file.write('                y = 255\n')
    file.write('            data[i][j][0] = int(y)\n')
    file.write('    data = cv2.cvtColor(data, cv2.COLOR_YUV2RGB)\n')
```

3) Results: From the comparison of Fig. 4a and Fig. 4b, we can see that the Strong Light Exposure Attack was successfully implemented. On initiation of the malicious change to the luminance, the monocular camera perception fails to identify the lane lines in the field of view. As a result, the victim AV veered off the lane onto the sidewalk, entering the off-road section of the driving environment. It lost perception and traversed the oncoming lane after being subjected to the Strong Light Exposure Attack. This immediately triggered the failure

TABLE I: Evaluation result of attack case 1.

Criterion	Result	Value
`RouteCompletionTest`	FAILURE	8.06 %
`OutsideRouteLanesTest`	FAILURE	11.79 %
`CollisionTest`	SUCCESS	0 times
`RunningRedLightTest`	SUCCESS	1 times
`RunningStopTest`	SUCCESS	0 times
`InRouteTest`	SUCCESS	
`AgentBlockedTest`	SUCCESS	
`Timeout`	SUCCESS	

TABLE II: Evaluation result of attack case 2.

Criterion	Result	Value
`RouteCompletionTest`	FAILURE	71.3 %
`OutsideRouteLanesTest`	SUCCESS	0 %
`CollisionTest`	SUCCESS	0 times
`RunningRedLightTest`	FAILURE	1 times
`RunningStopTest`	SUCCESS	0 times
`InRouteTest`	SUCCESS	
`AgentBlockedTest`	SUCCESS	
`Timeout`	FAILURE	

of the *Outside Route Test* and the *Route Completion Test*, terminating the simulation, as presented in Table I.

C. Attack Case II - Laser Beam

1) Attack Design: Adversarial machine learning (ML), as a form of cyber attack, involves designing a targeted numerical vector to make ML models misjudge and cause system failures and crashes. In this attack test case, the laser construction process is determined by several local-parameters: *intercept*, *injection Angle*, *wavelength*, and *laser width*. This laser attack is executed by randomly selecting a parameter and generating adversarial samples. If the confidence level of the classification is reduced, the current parameter settings are retained, which is often similar to the greedy strategy. After adding a laser beam projection to an image, the image pixels change, which in turn affected the results of the classifier. This adversarial attack, when applied to AD, can target the recognition of traffic lights, speed limit signs, and stop signs. Shining a laser on a stop sign can cause the AD system to fail to identify it correctly, leading to a violation of the required safety condition to stop the vehicle. Also, shining a laser on a traffic light can also create color spoofing attacks. Experimentation with laser beam attacks has shown that if the laser covers the entire traffic light, regardless of its color, the accuracy of detecting red or green lights is hardly affected. However, if the laser only shines on one traffic light, there will be a significant decrease in the recognition of the traffic light [12]. However, in our testing, we found that if we use this greedy strategy to search for the optimal parameters for 4000 cycles, the generation of adversarial samples is very slow, and it is impossible to inject adversarial samples into the AD test in real time. Therefore, we generate a laser that can make target recognition ineffective and recognise it as another object, by inputting images captured by the camera into an adversarial sample generation program. We then inject this laser in real-time in the AD test scenario. As in the previous case, we assume that the attacker can find appropriate attack scenarios and not be detected by others in advance. For example, the attacker can deploy multiple infrared light sources next to the road where the attacker's vehicle must pass or on a drone.

2) ADSecLang Configuration: The cyber threat scenario description using the ADSecLang is provided below.

```
attack_name: laser beam attack
attack_target: monocular camera-based end-to-end autonomous driving system
attack_goal: safety hazard
  action: [method, preconditions]
   method: [spoofing, shooting laser on the camera]
   preconditions: [{precond1, precond2, precond3}]
   precond1: [attack tool, laser pointer]
   precond2: [attacker's knowledge, machine learning adversarial sample generation technology]
   precond3: [attacker's capability, attackers can aim lasers at camera sensors on the roadside]
  impact: [influenced asset, influence]
   influenced_asset: [sensor, rgb_camera_front]
   influence: [raw_data, spoofed data]
```

The attack description YAML file is translated using the attack interpreter within the simulation platform.

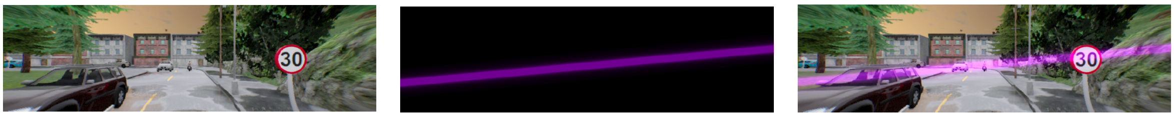
```
if(config['attack name']=="Laser beam attack"):
file.write('    laser_pattern = cv2.imread("laser_for_carriage.png")\n')
file.write('    if laser_pattern is None:\n')
file.write('        print("read image fail!!")\n')
file.write('        return 0\n')
file.write('    laser_pattern = cv2.cvtColor(laser_pattern, cv2.COLOR_BGR2RGB)\n')
file.write('    data = data.astype(np.float32)\n')
file.write('    laser_pattern = laser_pattern.astype(np.float32)\n')
file.write('    data = cv2.addWeighted(data, 1.0 , laser_pattern, 1.0 , 0)\n')
file.write('    data = np.clip(data, 0.0, 255.0).astype("uint8")\n')
```

3) Results: From the comparison of Fig. 5, we can see that the laser beam attack was successfully implemented in the AD simulation.

The attack achieved its objective of inducing AV behaviour to violate a safety condition. As shown in Table II, the vehicle completed approx. 70% of the route (*Route Completion Test*) and violated a safety condition by driving through a red light (*Running Red Light Test*). The result of the laser attack demonstrated that the laser beam was able to perturb the AD solutions perception of the traffic light, thus causing the victim AV to run a red light.

IV. Related Work

ADSecLang distinguishes itself from the state-of-the-art as it is the only domain-specific language, to our knowledge, for AD cybersecurity testing and it is designed to integrate within a software simulation testing environment for AD systems. Furthermore, the language has been designed to be agnostic to AD solutions or sensor technology and adaptable to accommodate diverse threat scenarios. SCENIC has been utilized to develop driving scenarios for cybersecurity testing. Salgado et al. [13]



(a) Before (b) Generated Laser (c) After

Fig. 5: Camera view of test case 2: (a) before the attack, (b) the generated laser beam, and (c) after applying the attack.

used the abstract and concrete scenario composition of SCENIC to create a scenario of a malicious leading vehicle in a convoy to test the robustness of cruise control and collision avoidance. This scenario demonstrates the effect if an attack had already succeeded, whereas the aim of ADSecLang is to incorporate the technical method of attack to assess the performance.

For more conventional threat modeling, *VehicleLang* [14] and *ALLIA (Agnostic Domain Specific Language for Implementing Attacks in an Automotive Use Case)* [15] are the two most prominent studies for legacy automotive architectures. Both of these solutions are focused on modeling cyber threats to connected vehicular systems and focus their case study evaluations on vehicular communication protocols and connected components. *VehicleLang* provides a conceptual contribution, which is the generation of text-based test cases whose feasibility can be validated by expert opinion. *ALLIA* extends this work by providing a technical implementation, which transforms the text-based test case generation into a technical test case implementation.

V. Conclusion

In this paper, we presented ADSecLang, a domain-specific language for autonomous vehicle cybersecurity testing. As part of the development of the language, we derived a taxonomy for AD cyber attacks and used the taxonomy to translate functional descriptions of threats to a domain-specific language that can be interpreted in the AD simulation testing platform. We demonstrated the feasibility and utility of ADSecLang to support a use-case evaluation of diverse attack scenarios on the camera sensor. The results demonstrated that ADSecLang was successful in generating attacks that could find vulnerabilities in a trajectory-guided end-to-end AD algorithm.

Future work for the development of ADSecLang will be to extend the language to encompass more diverse semantic cybersecurity scenarios and evaluate the utility of the language to support system-level attack scenarios (Buffer Overflow, Denial-of-Service, Network Attacks, etc.). We further aim to improve the results evaluation module. Metrics for AD testing predominantly focus on safety impacts, however, we would consider it necessary to define metrics that assist in directly evaluating the security of the system under test. Whilst this has proven a difficult challenge, the contemporaneous work on benchmarking for machine learning security and cybersecurity assurance levels (CALs) for automotive systems as conducted by the autonomous vehicle cybersecurity standardisation bodies provides some guidance how to achieve this. We further see the importance of integrating the language within a common AD cybersecurity testing evaluation platform, such as Simutack [16], for an open-source release.

Acknowledgment

This work is supported by the European Union-funded project CyberSecDome (Agreement No.: 101120779) and also co-funded by the European Union and Estonian Research Council via project TEM-TA5.

References

[1] R. S. Hallyburton, Y. Liu, Y. Cao, Z. M. Mao, and M. Pajic, "Security analysis of Camera-LiDAR fusion against Black-Box attacks on autonomous vehicles," in *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Aug. 2022.

[2] Y. Cao, S. H. Bhupathiraju, P. Naghavi, T. Sugawara, Z. M. Mao, and S. Rampazzi, "You can't see me: Physical removal attacks on lidar-based autonomous vehicles driving frameworks," in *Proceedings of the 32nd USENIX Conference on Security Symposium*, 2023.

[3] S. Kim, M. Liu, J. J. Rhee, Y. Jeon, Y. Kwon, and C. H. Kim, "Drivefuzz: Discovering autonomous driving bugs through driving quality-guided fuzzing," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CSS '22)*. ACM Press, 2022.

[4] Z. Wan, J. Shen, J. Chuang, X. Xia, J. Garcia, J. Ma, and Q. A. Chen, "Too Afraid to Drive: Systematic Discovery of Semantic DoS Vulnerability in Autonomous Driving Planning under Physical-World Attacks," in *Network and Distributed System Security (NDSS) Symposium*, 2022.

[5] D. J. Fremont, T. Dreossi, S. Ghosh, X. Yue, A. L. Sangiovanni-Vincentelli, and S. A. Seshia, "Scenic: A language for scenario specification and scene generation," in *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*. ACM Press, 2019.

[6] H. Chen, H. Ren, R. Li, G. Yang, and S. Ma, "Generating autonomous driving test scenarios based on openscenario," in *2022 9th International Conference on Dependable Systems and Their Applications (DSA)*, 2022.

[7] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive experimental analyses of automotive attack surfaces," in *Proceedings of the 20th USENIX Conference on Security*, 2011.

[8] H. S. Lallie, K. Debattista, and J. Bal, "A review of attack graph and attack tree visual syntax in cyber security," *Computer Science Review*, 2020.

[9] CARLA, "Carla autonomous driving leaderboard." [Online]. Available: https://leaderboard.carla.org/

[10] P. Wu, X. Jia, L. Chen, J. Yan, H. Li, and Y. Qiao, "Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline," in *NeurIPS*, 2022.

[11] C. Yan, W. Xu, and J. Liu, "Can you trust autonomous vehicles: Contactless attacks against sensors of self-driving vehicle," *Def Con*, 2016.

[12] R. Duan, X. Mao, A. K. Qin, Y. Chen, S. Ye, Y. He, and Y. Yang, "Adversarial laser beam: Effective physical-world attack to dnns in a blink," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, jun 2021, pp. 16 057–16 066. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR46437.2021.01580

[13] I. F. Salgado, N. Quijano, D. J. Fremont, and A. A. Cardenas, "Fuzzing malicious driving behavior to find vulnerabilities in collision avoidance systems," in *2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, 2022, pp. 368–375.

[14] S. Katsikeas, P. Johnsson, S. Hacks, and R. Lagerström, "Vehiclelang: A probabilistic modeling and simulation language for modern vehicle it infrastructures," *Computers & Security*, vol. 117, p. 102705, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404822001031

[15] C. Wolschke, S. Marksteiner, T. Braun, and M. Wolf, "An agnostic domain specific language for implementing attacks in an automotive use case," in *Proceedings of the 16th International Conference on Availability, Reliability and Security (ARES)*. ACM Press, 2021.

[16] A. Finkenzeller, A. Mathur, J. Lauinger, M. Hamad, and S. Steinhorst, "Simutack - an attack simulation framework for connected and autonomous vehicles," in *2023 IEEE 97th Vehicular Technology Conference (VTC2023-Spring)*, 2023.

Appendix IV

Paper IV

A. Roberts, M. Malayjerdi, M. Bellone, R. Sell, O. Maennel, M. Hamad, and S. Steinhorst. Analysis of autonomous driving software to low-level sensor cyber attacks. In 2025 IEEE/ACM 20th Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), pages 1–11, 2025.

Analysis of Autonomous Driving Software to Low-Level Sensor Cyber Attacks

Andrew Roberts*, Mohsen Malayjerdi†, Mauro Bellone*, Mert†,
Olaf Maennel‡, Mohammad Hamad§, Sebastian Steinhorst§
* FinEst Centre for Smart Cities, Tallinn University of Technology
† Department of Mechanical and Industrial Engineering, Tallinn University of Technology
‡School of Computer and Mathematical Sciences, The University of Adelaide
§Department of Computer Engineering, Technical University of Munich

***Abstract*—Autonomous Vehicle (AV) architectures fuse legacy, electromechanical components with advanced sensor technology and digital controllers, governed by software. An open challenge for the design of AVs are cyber threats such as Electromagnetic Injection (EMI) attacks, to the low-level layer, comprising electromechanical components, which can propagate through to the higher-level, intelligent control, affecting decision-making and the safety of the vehicle. This study analyses the robustness of the design of the software stack of a real-world AV to attacks on the low-level actuation using the example of an EMI attack on the steering angle sensor. To achieve this, we create a hybrid testbed which combines the mathematical model of the low-level sensor with the high-fidelity, intelligent control. We further develop safety and performance metrics measured at the high-level, which we use to generate a detailed view on the safety and system performance of the software. We conduct diverse EMI attacks on the target AV, within 3 diverse critical driving scenarios, consisting of $>$ 1000 simulations. The results indicate a correlation between an increase in attack noise with an increase in safety violations and failures to complete the mission of the AV. Our results highlight the importance for AV software developers of testing under diverse attack and driving scenarios, as each scenario within our experimentation exhibits different behaviour of the system and correlations to differing safety and system performance indicators.**

***Index Terms*—Security, Autonomous Driving**

I. Introduction

Cyber attacks that manipulate input to physical processes in cyber-physical systems present a fundamental challenge to secure system design [1]. Within the domain of automotive systems, transformation of legacy, analog architectures to digitally connected and autonomous driving (AD) technologies present new challenges. Legacy, analog automotive systems were designed based on a principle of contained, isolated system boundaries, restricting the flow of data within an analog system and sub-system [2]. The AD system architecture transforms this design, requiring the lower-level, analog control of actuation processes (steering control, braking, acceleration, etc.) to be open and connected to digital controllers so their process signals can be translated to digital input for the higher-level decision control [3].

There have been numerous real-world examples of semi-autonomous control architectures enacting unsafe decisions from erroneous sensing data from low-level actuation sensors [4] [5]. The 2018 SmartLynx Airline incident demonstrated that a physical disturbance from a maintenance activity on the horizontal stabilising sensor caused the sensing input to send erroneous data which propagated through to the control systems for flight planning, stabilisation and safety. The control systems initiated multiple concurrent actuation decisions (horizontal stabilisation, acceleration, etc.) which affected the safe operation of the flight [5]. Ultimately, manual intervention to override the autonomous control resolved the unsafe state of the flight.

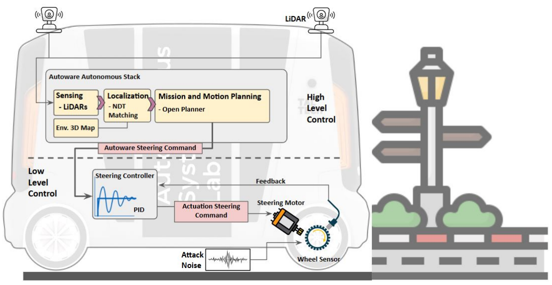

LiDAR
Autoware Autonomous Stack
Sensing - LiDARs
Localization - NDT Matching
Mission and Motion Planning - Open Planner
Env. 3D Map
High Level Control
Autoware Steering Command
Low Level Control
Steering Controller
PID
Feedback
Steering Motor
Actuation Steering Command
Attack Noise
Wheel Sensor

Fig. 1: High-level architecture of Steering Angle Sensor Manipulation within AD System.

Within the context of cyber threats, numerous studies have proven the vulnerability of microelectronic sensors to electromagnetic interference (EMI) [6], [7], [8], acoustic sensor [9] [10] and data manipulation attacks [11], [12], [13], [14]. Furthermore, the network that exchanges actuation signals, the Control Area Network (CAN) Bus network, has been shown to be inherently vulnerable to a diversity of man-in-the-middle [15], [16] attacks. Yet, there is a lack of practical investigation that extends this analysis of the propagation of malicious data input within an AD system, where physical processes are software-controlled and manual, and human intervention is not available.

Our study is motivated to investigate how cyber attacks to electromechanical components, in our case, a steering-angle sensor, propagate through the AV system, affecting higher-level decision-making. The aim of the study is to analyse the design of a real-world AD vehicular system and assess mechanisms to enhance the design of the architecture of AD systems to be more robust and resilient. To achieve this, we, investigate a real-world AV software ecosystem, analysing the integration between the lower-level control, characterised by electromechanical components, and the high-level control, characterised by digital systems which support algorithmic decision-making. We then investigate how malicious input propagates within this ecosystem. Finally, we determine mechanisms for enhancing secure design.

To guide this research, we focus on the following research questions:

RQ1 How does a manipulation to the electromechanical component propagate through the AD software stack?

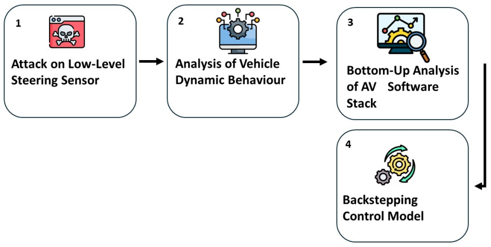

1
Attack on Low-Level Steering Sensor
2
Analysis of Vehicle Dynamic Behaviour
3
Bottom-Up Analysis of AV Software Stack
4
Backstepping Control Model

Fig. 2: Conceptualization of our approach, from attack to backstepping.

RQ2 *What dependencies exist between the AD control algorithm and low-level control?*

RQ3 *Where in the architecture of the autonomous vehicle can defensive mechanisms be placed to defend against control invariants?*

Contribution:

- This is the first study, to the best of our knowledge, that conducts an analysis of a production, full-autonomy (Level 4-5), AV system to cyber attacks to the lower-level control.
- Amidst the pressing need to address the design of secure digital critical-infrastructure systems in transportation, we analyse the relationship between low-level, electromechanical systems and high-level control software within AV systems.
- The study provides recommendations for enhancing robustness and resiliency in the design of AV software.

II. Approach Overview

Our approach (see Fig. 2) is to, *first*, implement the sensor interference attack model in our custom high-fidelity AD test-bed environment. The test-bed environment contains the software stack of our real-world vehicle and configurations consistent with the real-world kinematics of the vehicle.

Second, from the results of the experiments, we assess the impact of the cyber attacks utilising defined safety criteria. Furthermore, we conduct a sensitivity analysis of the vehicle's dynamic parameters to identify the behavioural effect of the malicious input and assist in pinpointing critical areas of the AV software which are affected by the attack.

Thirdly, we conduct a bottom-up analysis, to ascertain what happens to the high-level, decision-control, when malicious data is injected into the low-level. The bottom-up analysis details the relationship between inputs and outputs in the AV software stack.

Fourth, the previous analysis enables backstepping at a conceptual level to stabilize elements of the control model that are susceptible to the sensor interference attack.

We justify the use of this approach as it enables us to take an architectural view of the AV software stack. Existing studies use methods that view the problem of manipulation of low-level sensor input either within the context of a PID control [6] [7] issue or solely focus on the autonomous control [12]. We believe that taking an architectural perspective, where the interconnections and dependencies of the system are encountered, enables the designer/s of the AV to gain more insight into the functioning of the system under attack.

III. Autonomous Vehicle Software Architecture

The aim of this section is to describe the entire software stack of the autonomous vehicle platform from a bottom-up perspective. Each layer of the stack will be detailed as to there purpose/task and how they communicate.

A. Low-Level Control

Low-level control is at the base of our software stack, having the task of giving actuators the right commands to generate a desired behaviour. Analog controllers have the function to follow a specific reference signal. It is clear that such signals are measured by transducers and applied to actuators as current or voltage signals to apply a torque to a motor at the low level. The most common and well known analog controller in automotive is the Engine Control Unit (ECU), which regulates injection, speed, and other engine parameters. Brake control modules are also very common and control various aspects of the braking system, such as anti-lock braking (ABS), electronic stability control (ESC), traction control, and brake force distribution. Now, assuming that our goal is to keep any value of cruise speed, a velocity regulator works by receiving a measure of the current speed, comparing it to the reference, and generating a control signal to accelerate or brake accordingly. Low-level controllers typically work on a simple control feedback loop involving some type of linear system model (or a linearized one). The most common, state-of-the-art, and well-established controller in automotive is the PID (Proportional-Integral-Derivative) controller. They are widespread in automotive for their simplicity, robustness, usability, and real-time capabilities. A PID controller continuously calculates an error signal based on the difference between a desired setpoint and the measured process variable and then adjusts the control output accordingly. They use proportional, integral, and derivative actions to regulate a vehicle's actions. The underlying equation is relatively easy, involving three constants, proportional, integral, and derivative constants, typically indicated as K_p, K_i, and K_d, to weigh each action respectively. With reference to Fig. 3, PID controllers are at the base of the "drive controller" and "steering controller" block.

Control theory provides a very stable mathematical theory about analysis and synthesis of the controllers, thus how disturbance might affect the controller is, in principle, well known. This work aims to provide insights on how the behaviour of the controller might affect the decision-making blocks in a real-world, operational AV.

B. Intermediate Layer/Master Controller

The role of the Master Controller is to parse analog input to the digital network of the vehicle.

The Master Controller communicates with the low-level control through the CAN bus. The low-level control section in Fig. 3 shows all the basic components in our system, which are connected to the master controller by three different CAN busses:

- CAN1 is used to connect all safety-critical components, such as brake systems and electric motors.

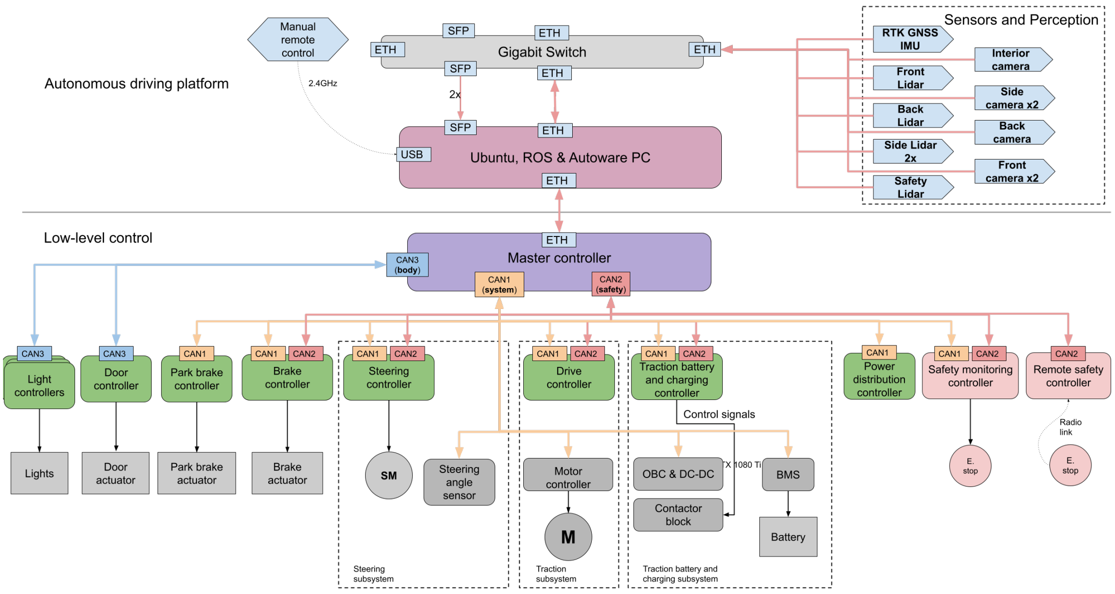


Fig. 3: Autonomous vehicle high-level functional architecture.

- CAN2 is used for redundancy over all the safety-critical components.
- CAN3 is dedicated to low-priority body-related functions such as door automation and lighting.

The master controller receives data from the low level via CAN bus and forwards to the upper-layers via ethernet. Then receives processed signals from the intelligent blocks (the upper-layers) and generate the control commands for the actuators, parsed via CAN bus. Basic data from low-level sensors are processed here and forwarded to the upper layer, this includes speed, acceleration, encoder positioning, voltage and currents.

What this means is that an anomaly in the PID controller, or in its feedback loop, is read here and propagate to the upper layer as limited computation (ARM 32-bit Cortex M7 CPU) can be done at this level. The master controller directly communicates with the upper levels (i.e. Autonomous Driving Software) via ROS topics flowing over the ethernet connected to the Ubuntu-based Autoware PC.

C. Autonomous Driving Platform

The real-world AD platform is fully implemented on a PC featuring an AMD Ryzen Threadripper 1950X (16-core/32-thread) CPU, 2 NVIDIA GeForce GTX 1080 Ti graphic cards, and 64 GB of memory. The driving platform runs the Autoware stack [17], over a ROS environment in the Ubuntu 18.04 OS. Here is where the ROS master is run, handling all topics/subscribers from/to the low level and from the sensing level, but also to each algorithmic component running on concurrent threads and consuming the incoming data flow while providing interpreted information. The intelligent driving software stack includes several modules interpreting information from bottom-up and top-down, referring to high-level perception and low-level control. The main task is to coordinate concurrent processes of sensing, localization, and planning.

The important task of planning and decision-making is assigned to the Open-Planner algorithm [18], which generates a waypoint according to the information coming from the sensors and forward this information to an intelligent control block that generates the actual trajectory in consideration of kinematic/dynamic model of our system. The planning algorithm also generates alternative paths referred to as ”rollouts” which serve as possible drivable trajectories. Each rollout is a kinematically compliant path on which the vehicle could be driving. The change from one rollout to another might normally occur in case of maneuvers (such as overtaking, obstacle avoidance, and intersection crossing), to optimize energy efficiency, or in case of violation of any safety criteria.

This unit is ultimately responsible for the intelligent control in our autonomous vehicle, from localization to planning, perception and control.

Differently, with respect to the low-level controllers (such as PIDs) that are limited to follow a reference point, the intelligent controller decides the reference point to be sent to the low level, and resets it at run time according to the real behaviour of the vehicle. This block can be interpreted as a wider feedback loop acting on a macro scale receiving information from the environment and from the low level. It is clear that, on a micro-scale, the PID controllers are effective, and provide mathematical guarantees of convergence and robustness to noise, while on a macro scale intelligent controllers cannot provide similar guarantees. In case of attack, misleading information, or any source of uncertainty, intelli-

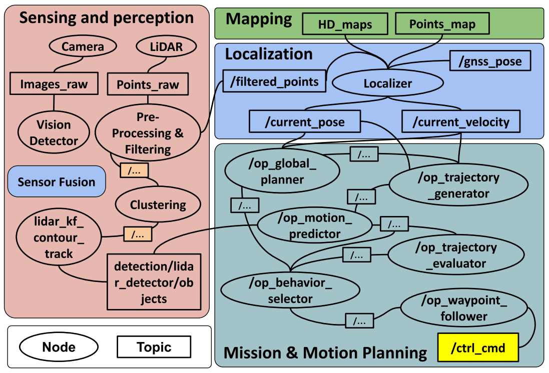


Fig. 4: Intelligent driving software stack structure showing ROS nodes/topics communication between essential elements.

TABLE I: Autonomous Vehicle Sensors.

Sensor	Model
3D lidar (front)	Velodyne Ultra puck VLP 32
3D lidar (rear)	Velodyne VLP-16
2xSide lidar	Robosense Bpearl
Safety lidar	Ouster OS0-90 (Safety)
3xCamera	Flir
GNSS	Trimble BX992
Radar	TI

gent controllers can generate catastrophic decisions, thus our goal is to better understand how uncertainties from the low-level can propagate to the high level generating wrong driving actions.

D. *Intelligent driving software stack*

A part of the ROS nodes/topics running on the vehicle are represented in Fig. 4. The software stack is mainly composed of the following main components, sensing and perception, mapping, localization and motion planning. Perception modules runs AI-based modules for detection, segmentation and interpretation of traffic scenes. Localization and mission planning receive constant feedback from vehicle and global positioning to generate new control commands. We expect that our attack, though not directly carried out on those modules, would generate errors that propagate from the low level to the localizer and trajectory-generator blocks.

1) Sensors: Sensors are connected to the AD platform running AI-based models for identification, detection and segmentation of objects and environmental information through a Gigabit ethernet switch. Data flow is managed and synchronized directly in the Autoware stack, sending data as ROS topics to concurrent threads (nodes) running inference over the AI-based deployed modules. Sensing information are used for perception-related functionalities such as object detection, segmentation and sensor fusion. Table I provides the list of available sensors.

IV. Adversarial Model

The objective of the attacker is to cause the AV to take unsafe driving actions resulting from manipulation of the steering angle sensor.

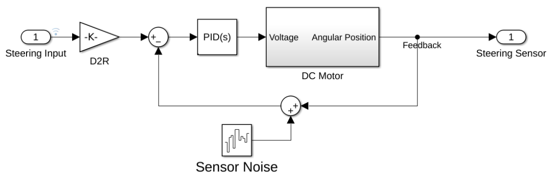


Fig. 5: Steering angle sensor attack.

We assume the attackers cannot directly access the digitised sensor readings. Instead, we assume that the attacker can exploit vulnerabilities in the steering angle sensor using proven techniques such as EMI, to affect the integrity of the sensor data (analog signals on the signal conditioning path before being digitised).

We assume that the attackers can physically place an EMI device near the steering angle sensor and are capable of crafting and transmitting interference to the sensor during the navigation of the AV and thus transform the waveform of the sensor output. We further assume that the attackers do not possess an in-depth understanding of the voltage levels of the steering sensor and therefore focus on injecting incremental noise into the sensor.

We assume that the attackers can observe the operation of the AV and control the attack in terms of initiation and cessation of the attack during varied time periods or within the frames of a critical driving manoeuvre.

V. Attack Model

The attack is conducted in the measurement of the input and output of the PID controller for the steering angle (Fig. 5). The key parameters that affect the success rate of the attack are: *duration*, *noise*, *attack trigger action*.

Within our attack model, attacks are conducted with differing sensitivity levels of the steering angle sensor and durations and are triggered at targeted points of the AV mission. We have chosen a range of sensor attack noise levels (0.01, 0.05, 0.1, 0.2), rather than a specific target point. The study of Pöllny et al. [6], which conducted EMI attacks on a sensor used in an automotive electronic control unit (ECU), indicated that an attacker does not need to set a specific value for the steering angle attack, but simply to find the sufficiently high level of noise that would alter system behaviour to the attacker goal.

Whilst, EMI attacks have been proven successful against microelectronic components in [19] [12] [6] [11] [7] [8] [10], the attacks are applied to the stand-alone sensor hardware and application use-cases such as microphones, temperature sensors, drones. The novelty of the attack model in our study is the implementation of the attack to a fully-autonomous vehicle that integrates low-level actuation with high-level AD decision-making. This enables the ability to assess the affect of the attack on the entire AV software stack. Furthermore, the attack is conducted utilising scenario-specific testing. This is of critical importance, as it is widely understood that the performance of the AD decision-making layer differs based on scenario-specific behaviour [12]. For the AD algorithms may be better optimised for specific driving manoeuvres such as overtaking, or operational driving domains (ODD) such as busy intersections. Our attack is conducted in a simulation test environment, as attacks at the physical, hardware-level are proven, the gap in existing research, is how these inputs propagate within the system and affect the decsion-making within an autonomous system.

VI. Experiment

A. Experimental Setup

To conduct the attack and analyse the subsequent effects, we developed an experimental test environment.This environment consists of a simulation platform that fuses the low-level actuation, simulated in MATLAB, with a high-fidelity simulation of the AV software of our real-world vehicle, simulated in CARLA. The simulation test environment provides an optimal platform as it uses the same mathematical model of the steering actuation sensor and the same software stack as the real-world vehicle. Furthermore, the simulation environment enables attack testing to be conducted in an agile manner, whilst, removing the safety risk factors of testing the AV in the physical road environment.

B. Attack Implementation

We chose to conduct the low-level attack on three diverse scenarios (see Fig. 6): 1) Straight-line, 2) Overtaking manoeuvre and, 3) Left-turning maneuver at intersection. These scenarios were chosen as they are consistent with the most-popularly tested driving scenarios according to the survey of test methods and practices in [20]. As shown in Fig. 6, the high-fidelity simulation view for the three scenarios is conducted. The Straight-Line scenario shows that the EMI attack is initiated after the vehicle traveled 20 meters, with two different attack durations: 10 and 20 meters. For the overtaking manoeuvre, the attack begins during the cut-in process and lasts for 10 meters. Finally, in the intersection scenario, the attack is launched as the vehicle enters the intersection and persists for a distance of 10 meters.

To conduct our experiments, firstly, we conduct the scenario with *no-attack* for 100 runs. This establishes a baseline of the performance of the AV without attacks. From there, each of the attacks with different noise levels and duration are run 100 times. Overall, approx. 1900 simulation runs are recorded, and as the high-fidelity simulation uses GPU and CPU resources, this is a time-consuming process. Figure 7 presents the scenario flow used to integrate the attack into the mission in CARLA. It outlines the sequence of behaviors from the vehicle's initialization and driving towards the goal to executing an attack or stopping based on a distance trigger. The attack is enabled based on a predefined condition. This structured flow allows for precise control over when and how the attack occurs during the scenario, ensuring consistent testing of the AV's response to disturbances.

C. Evaluation Criteria

Table II and III detail the safety and performance criteria applied in our experiments, respectively. As we have diverse scenarios which involve scenarios with ego vehicles, certain criteria is only applicable to their corresponding scenario. In this analysis, mission failure (NotF) and safety violations (SafetyV) are distinct evaluation criteria used to assess the performance and safety of the AV during the scenarios.

Mission failure (NotF) refers to instances where the vehicle was unable to complete the mission. This typically occurs

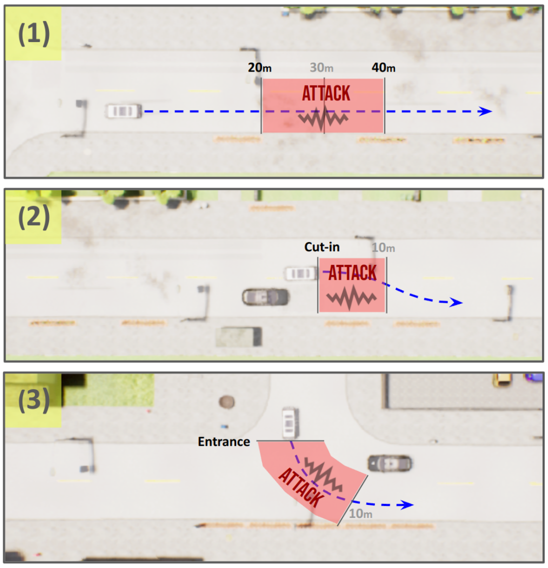


Fig. 6: Game-engine view of three simulated scenarios representing the attack occurrence place during the mission; 1) Straight-line 2) Overtake 3) Intersection.

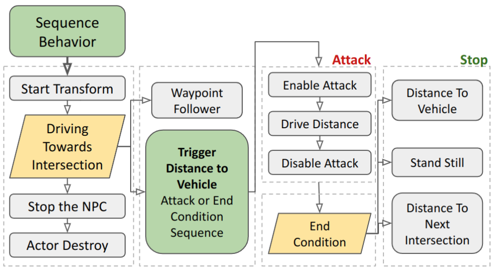


Fig. 7: Flow graph of how each scenario is processed in the simulation platform.

due to critical events that prevent the AV from finishing its task, such as collisions (V_{Col}), localization loss (V_{NDTLs}), or sidewalk incursions (V_{SiIn}). These violations are severe enough to terminate the mission.

Safety violations (SafetyV), on the other hand, refer to any breaches of safety that occur during the mission but do not necessarily prevent the vehicle from completing it. A mission may still be considered successful even if multiple safety violations are recorded. Examples of these include deviation to the center lane (V_{DTL}), sharp braking (V_{BrD}), localization loss (V_{NDTLs}), collisions (V_{Col}), and violations of distance to collision (V_{DTC}VDTC). In these cases, while the AV may exhibit unsafe behaviors or suboptimal performance, it is still able to complete the mission.

Two critical safety metrics are sidewalk incursions (V_{SiIn}) and collisions (V_{Col}), both representing severe safety hazards. A sidewalk incursion indicates where the AV veered off its

TABLE II: Safety Evaluation Criteria.

Safety Condition	**Data Label**	**Description**	**Metric**
Not Finished	*NotF*	Failure to finish the mission	Pass/Fail
Sidewalk Incursion	*SiIn*	AV deviation into pedestrian zone	Pass/Fail
Collision	*Col*	AV collides with NPC	Pass/Fail
Distance-to -Collision	*DTC*	Violation of the safe distance between AV and NPC	AV within 0.5m of other vehicle
Distance-to -Centre Lane	*DTL*	Violation of the safe distance between AV and Centre Lane	AV within 0.4m of centre lane
Break on Driving Lane	*BrD*	AV initiates emergency break on driving lane	Pass/Fail
Localization	*NDTLs*	Localization Loss	NDTerror > 1.0
Violation	*V*	Safety Violation	

TABLE III: Performance Evaluation Criteria.

Performance Criteria	**Data Label**	**Description**	**Metric**
Lane Transition	*RlOut*	AV executes multiple roll-out transition	Pass/Fail
Localization	*NDT*	Localization Performance	AV localization matching
Localization	*NDTer*	Mean localization pose error	Localization error margin
Duration	Dur	Duration in seconds	
Max NDT score	MxNDTSr	Max NDT score during a mission	Smaller = Better
Path Deviation	Dev2Ref	Sum of deviation to the reference path in sampled points	Smaller = Better
Max Lat Deviation	MxLaDev	Max lateral deviation from original path	Smaller = Better

intended path and encroached into pedestrian zones, potentially endangering people on sidewalks. Similarly, a collision signifies an event where the AV collided with a nearby non-player character (NPC) vehicle.

Another key performance indicator is the deviation to the reference path (Dev2Ref), which measures how far the AV strayed from its intended trajectory. It is important to note that Dev2Ref is not the deviation at a single point; rather, it represents the summation of the deviations at several reference points along the planned path to the actual route traveled by the AV. This cumulative nature of the metric results in larger values, especially when the AV frequently deviates from the intended trajectory.

VII. Results

For each of the scenario's, the results, as expressed in Tables. IV, V, and VI demonstrate that increasing level of noise and duration of the EMI attack impact the safety and performance of the AV.

The manipulation of the steering sensor input at higher noise levels affects the feedback loop for the calculation of localisation, which results in the AV experiencing loss and jumps of localisation. The NDT algorithm, used in the localisation algorithm, exhibits weakness in holding the position of the AV during sensor manipulation, which is demonstrated by loss of localisation, in attempting to re-correct, it incurs jumps. The loss and jumps of the localization affect the displacement of the AV. As such, the cost-based algorithm used by the mission and motion planning module recalculates the trajectories and chooses a new roll-out. The choice of a new trajectory for the AV disrupts the flow of critical maneuvers within the scenario, such as the cut-in process of overtaking,

TABLE IV: Summary of the Safety and Performance Evaluation - Straight Line Scenario. The first line is our baseline path, where no attack was applied.

SAFETY							
Length	Noise	NotF	SafetyV	V_{SiIn}	V_{DTL}	V_{NDTLs}	V_{BrD}
-	baseline	0%	0%	0%	0%	0%	0%
10 m	0.01	0%	0%	0%	0%	0%	0%
10 m	0.05	10%	10%	0%	10%	0%	0%
10 m	0.1	12%	12%	0%	6%	6%	0%
10 m	0.2	30%	30%	2%	26%	12%	8%
20 m	0.01	0%	0%	0%	0%	0%	0%
20 m	0.05	34%	34%	2%	30%	8%	2%
20 m	0.1	34%	36%	4%	28%	18%	6%
20 m	0.2	42%	42%	6%	38%	14%	2%

PERFORMANCE					
Length	Noise	Dur	RlOut	MxLaDev	MxNDTSr
-	baseline	57.6s	0	0.1m	11.9
10 m	0.01	59.9s	0	0.2m	11.9
10 m	0.05	61.5s	0.16	1.6m	12.0
10 m	0.1	65.5s	0.3	1.5m	12.5
10 m	0.2	71.8s	1.18	8.3m	25.5
20 m	0.01	70.2s	0	0.3m	14.2
20 m	0.05	75.6s	0.94	1.7m	25.5
20 m	0.1	82.6s	1.36	8.2m	46.9
20 m	0.2	85.6s	1.64	8.2m	35.0

smoothing of trajectory in keeping straight-line and turning at the intersection.

A. Scenario 1: Straight-Line

Within the Straight-Line Scenario Safety Results (Table. IV), safety violations begin to occur when 0.05 noise is introduced into the sensor input, marking the threshold where the AV system starts to struggle with maintaining safety. At this noise level, a 10% safety violation rate provided by lateral deviation violations was observed. As the noise level and attack duration increase, the AV experiences a progressive degradation in performance, culminating in the highest noise level (0.2) and the longest attack duration (20 meters), which results in a 42% safety violation rate and 38% lateral deviation violation.

A key characteristic of the AV's behavior in this scenario is the Deviation-to-Centre-Lane. The noise is injected into the steering sensor, and abrupt changes in the steering actuation cause the vehicle's control system to oscillate between making corrections and following the desired path. Autoware's motion planner attempts to rectify the vehicle's course, but the corrections are often sub-optimal, resulting in the AV veering to a dangerous proximity to the center line. This behavior indicates a weakness in the resilience of the AV's planning algorithm when recovering from anomalous inputs, as the system fails to regain optimal performance after the attack.

A more extreme example of dangerous trajectories, is where the EMI injection causes the AV to lose localisation which, cascades to affect the decision-making of the planning algorithm. The attack localization loss, as indicated by the NDT Error Value and NDT Score increasing, and the sharp variances between autoware and simulator. This behaviour results in the AV veering into the adjacent lane and hitting the side curb, a behaviour characteristic of 6% of the runs within the maximum noise and duration simulation set. Associated with these safety violations are significant performance degradation. In scenarios with low noise levels (0.01), the maximum lateral

TABLE V: Summary of the Safety and Performance Evaluation - Overtake Scenario. No attack was carried out in the baseline experiment.

SAFETY							
Noise	NotF	SafetyV	V_{SiIn}	V_{Col}	V_{NDTLs}	V_{DTC}	V_{BrD}
baseline	0%	1%	0%	0%	0%	1%	0%
0.01	7%	18%	2%	3%	4%	14%	1%
0.05	16%	23%	8%	3%	11%	10%	2%
0.1	29%	40%	18%	2%	26%	14%	1%
0.2	33%	39%	23%	7%	28%	14%	2%

PERFORMANCE						
Noise	Dur	RlOut	$\overline{DTC}$	MxNDTSr	NDTer	S-NDTer
baseline	104.7s	8.2	0.4m	19.4	0.2m	0.1m
0.01	107.3s	7.8	0.2m	55.9	0.2m	0.2m
0.05	121.4s	8.9	0.2m	73.9	0.4m	0.5m
0.1	125.4s	10.0	0.2m	63.9	0.7m	0.9m
0.2	124.7s	10.2	0.2m	53.1	0.6m	0.8m

deviation is limited to around 0.2 meters. However, under maximum noise (0.2) and 20-meter duration conditions, the lateral deviation increases dramatically to 8.2 meters, showcasing the substantial impact of noise on the AV's ability to maintain its path. This severe lateral deviation illustrates the danger posed by noise-induced errors in the vehicle's steering and localization systems.

Moreover, the *RlOut* metric—which tracks the average number of local trajectory transitions during a mission—shows a significant increase under high-noise conditions. This indicates the motion planner's growing uncertainty and inability to maintain a stable trajectory. As the AV continuously switches between trajectories, it struggles to converge on an optimal path, leading to erratic driving behavior and further deviations. Another factor exacerbating these challenges is the increased mission duration under noise attacks. The AV, displaced from its efficient path due to trajectory deviations and localization errors, takes longer to complete the mission. In the 0.2 noise / 20-meter scenario, the mission duration extended by nearly 28 seconds compared to the no-attack baseline, reflecting the inefficiency introduced by the noise attacks.

B. Scenario 2: Overtake Maneuver

In this experiment, the attack length was fixed at 10 meters while varying the noise levels to assess their impact on the vehicle's performance and safety. In the no-attack scenario (see Table. IV), the AV successfully completed the overtaking maneuver with minimal disruptions. The mission failure rate (NotF) was 0%, and a 1% violation of distance to collision (V_{DTC}) was recorded, indicating that in one case, the vehicle exceeded the safe distance from nearby objects. Despite this, there were no sidewalk incursions (V_{SiIn}), collisions (V_{Col}), or localization loss (V_{NDTLs}). The vehicle maintained a safe average DTC of 0.4 meters. The mission duration was 104.7 seconds, with an NDT error of 0.2 and a standard deviation of 0.1.

In the 0.01 noise scenario, V_{NotF} increased to 7%, and by the 0.2 noise level, it reached 33%. Similarly, $V_{NDTLoss}$ was first observed at 0.01 noise (4%), growing to 28% in the 0.2 noise scenario. These results indicate that noise in the sensor input significantly disrupts the vehicle's ability to maintain accurate localization, directly impacting mission success.

In the no-attack scenario, V_{SiIn} and V_{Col} were recorded at 0%, reflecting ideal behavior where the AV stayed within its designated path and successfully avoided NPCs during overtaking. However, as noise levels increased, both metrics worsened. In the 0.01 noise scenario, V_{SiIn} rose to 2%, and V_{Col} to 3%, showing the system's diminished capacity to maintain lane discipline and avoid nearby vehicles. At the highest noise level (0.2), sidewalk incursions increased to 23%, while collisions reached 7%, a significant rise indicating the AV's inability to safely manage the overtaking maneuver under heavy noise interference. These results suggest that sensor noise not only disrupts the vehicle's path but also critically impacts its ability to avoid hazards that could lead to severe accidents involving both pedestrians and other vehicles.

The V_{DTC}, which reflects the rate at which the AV exceeded safe distances from nearby objects, increased from 1% in the no-attack case to 14% in the 0.2 noise scenario. This was accompanied by a rise in sharp braking events as the AV's control system struggled to compensate for the noisy input, leading to more frequent sudden stops. As the noise level increased, the RollOut metric showed greater instability. In the 0.2 noise case, the RollOut metric increased from 8.2 (in the no-attack scenario) to 10.2, indicating the planner's increasing uncertainty in maintaining a stable trajectory.

The mission duration increased as the noise level rose. In the 0.2 noise scenario, the AV took 124.7 seconds to complete the maneuver, an increase from 104.7 seconds in the no-attack scenario. Additionally, the NDT error and its standard deviation saw significant increases, with the NDTer rising from 0.2 to 0.6 and the S-NDTer increasing from 0.1 to 0.8, highlighting the degradation in localization performance under noisy conditions.

C. Scenario 3: Intersection

In the intersection scenario, the attack length remained unchanged at 10 m, while the noise levels varied to assess their impact on the AV's performance during this complex maneuver. In the baseline scenario, the AV successfully navigated the intersection without mission failure (0%) or significant safety violations, aside from a small 3% V_{DTC}. There were no recorded V_{SiIn} or V_{Col}, and the AV maintained an average DTC of 0.4 meters, with an NDTer of 0.1 and a minimal deviation from the reference path of 20.4 meters. The overall mission duration was 65.8 seconds, and the system performed with only 2.2 RollOut changes, indicating a stable and efficient planning process.

As noise levels increased, the NotF rate rose from 8% at 0.01 noise to 25% at 0.2 noise. Safety violations also saw a sharp increase, particularly in terms of V_{NDTLs}, which jumped from 7% at 0.01 noise to 22% at 0.2 noise. This degradation in localization directly impacted the AV's ability to make timely decisions and follow the intended trajectory, leading to more dangerous driving behavior.

While sidewalk incursions and collisions were rare in the baseline scenario, they became more frequent as noise levels rose. At 0.2 noise, 4% of the runs resulted in V_{SiIn}, and 4% in V_{Col} with non-player characters (NPCs) within the intersection. This behavior indicates a critical safety failure, where the AV not only lost control of its lane discipline but also failed to avoid NPCs and pedestrian zones.

TABLE VI: Summary of the Safety and Performance Evaluation - Intersection Scenario. No attack was carried out in the baseline experiment.

SAFETY							
Noise	NotF	SafetyV	V_{SiIn}	V_{Col}	V_{NDTLoss}	V_{DTC}	$\overline{DTC}$
baseline	0%	3%	0%	0%	0%	3%	0.4m
0.01	8%	15%	0%	1%	7%	10%	0.2m
0.05	19%	27%	2%	3%	16%	13%	0.2m
0.1	23%	32%	6%	3%	19%	16%	0.2m
0.2	25%	28%	4%	4%	22%	7%	0.1m

PERFORMANCE							
Noise	Dur	RlOut	MxNDTSr	NDTer	S-NDTer	Dev2Ref	S-Dev2Ref
baseline	65.8s	2.2	38.5	0.1m	0.1m	20.4m	8.2m
0.01	70.5s	3.1	39.5	0.2m	0.2m	39.1m	98.8m
0.05	72.9s	3.9	40.9	0.4m	0.4m	63.6m	170.4m
0.1	74.2s	4.5	37.5	0.5m	0.5m	69.6m	147.5m
0.2	74.5s	4.1	39.1	0.4m	0.5m	77.9m	154.9m

The cumulative deviation remained relatively low in the no-attack baseline scenario, indicating stable performance. However, under the influence of noise, this deviation increased significantly. For example, in the 0.2 noise scenario, the Dev2Ref reached 77.9 meters, with a high standard deviation of 154.9 meters, demonstrating the system's growing instability under attack. The high standard deviation reflects the inconsistency in the AV's ability to maintain a predictable trajectory, as deviations varied considerably at different points along the path. The increasing Dev2Ref values show that the AV struggled to recover from noise-induced errors, leading to significant drift from the planned path.

The results show that the roll-out metric increased as noise levels rose. In the 0.01 noise scenario, the roll-out increased to 3.1, and by 0.2 noise, it rose to 4.1, indicating the planning system's growing uncertainty in selecting and maintaining a stable path. The maximum NDT score also fluctuated, reaching a high of 40.9 in the 0.05 noise scenario, highlighting the deteriorating localization performance.

The NDT error and its standard deviation also increased with higher noise levels. At 0.2 noise, the NDT error rose to 0.4, with a standard deviation of 0.5, indicating significant localization drift. This localization instability contributed to unsafe driving behavior, as reflected in the increased V_{DTC} and collisions. The mission duration also increased with noise levels, from 65.8 seconds in the baseline scenario to 74.5 seconds at 0.2 noise. This duration increase indicates the AV's struggle to efficiently navigate the intersection under attack, as the planning algorithm and control systems were frequently forced to adjust to counteract the noise-induced deviations.

D. Comparison Between Safety Violations and Simulated Scenario

Figure 8 represents radar graphs that provide a clear visual representation of the impact of noise attacks on the AV across all different mission types: straight-line driving, overtaking, and intersection maneuvers, with varying attack lengths (10 meters and 20 meters) for the straight-line scenario. By comparing these radar graphs, we can discern how the attack influences the AV in different maneuvers and understand whether the vulnerability is related to the nature of each maneuver.

In the straight-line scenario (Fig. 8 (a) and (b)), the radar plots show a clear difference between the 10-meter and 20-

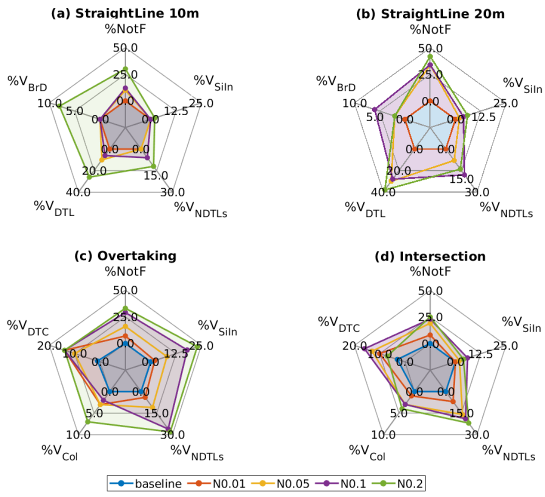


Fig. 8: Safety violation of simulated scenarios.

meter attack lengths. With the 10-meter attack (Figure (a)), the V_{DTL} and V_{NDTLs} are relatively contained at noise levels below 0.1, but they spike at 0.2 noise, indicating that longer attack lengths exacerbate the vehicle's struggle to maintain its trajectory. By contrast, in the 20-meter attack scenario (Figure (b)), the impact of noise is more pronounced across all noise levels, with a higher percentage of NotF and significantly greater V_{DTL} and V_{NDTLs} values. This suggests that the longer attack duration amplifies the system's inability to recover from perturbations in the steering sensor, causing the AV to deviate further from the planned path.

In the overtaking scenario (Fig. 8 (c)), the radar plot highlights that this maneuver is particularly vulnerable to V_{NDTLs} and V_{DTC} as noise levels increase. Even at 0.01 noise, the AV shows a marked increase in these safety violations, and by 0.2 noise, V_{NDTLs} and V_{DTC} reach critical levels. This indicates that overtaking is a more complex and challenging maneuver for the AV compared to straight-line driving, as it requires the vehicle to safely execute lane changes and avoid collisions with NPCs. The complexity of coordinating between localization, path planning, and collision avoidance makes the system more prone to safety violations when noise is introduced.

In the intersection scenario (Fig. 8 (d)), the radar plot demonstrates that this maneuver is less affected by V_{DTC} compared to the overtaking scenario, but the mission failure rate and localization loss are notably higher. Even at 0.01 noise, NotF jumps to 8%, and V_{NDTLs} reaches 7%, while at 0.2 noise, NotF reaches 25%, indicating a substantial failure rate. The intersection maneuver places a high demand on the AV's localization and planning systems, as it requires precise decision-making in a constrained environment with multiple potential collision points. The increase in safety violations with rising noise levels reflects the difficulty the AV faces in maintaining control during complex navigation tasks in intersections, where it must simultaneously monitor multiple

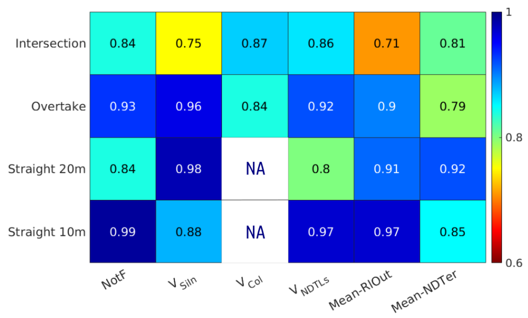


Fig. 9: Correlation coefficients between violation metrics (horizontal axis) and noise levels ([0, 0.01, 0.05, 0.1, 0.2]) for each scenario (vertical axis). The values indicate the strength of the relationship between the likelihood of each violation and changes in noise levels.

potential threats and adjust its trajectory.

The vulnerability of the AV to noise attacks appears closely tied to the nature of the maneuver. Straight-line driving is less demanding in terms of control and localization, and as a result, the AV is able to handle noise better—though longer attack durations (as in Fig. 8 (b)) significantly increase the risk of mission failure. In contrast, overtaking involves more dynamic path changes and collision avoidance, making it more susceptible to noise, as seen in the sharp rise in V_{DTC} and V_{NDTLs} even at low noise levels. Intersection maneuvers also present significant challenges, particularly due to the need for precise localization and decision-making at multiple points, resulting in higher mission failure rates and localization loss as noise levels increase. These findings suggest that the more complex the maneuver (i.e., those requiring more dynamic control and interaction with external factors like NPCs or intersection points), the more vulnerable the AV is to noise attacks.

E. *Violation to noise correlation analysis*

The correlation heatmap shown in Fig. 9 reveals significant insights into how different safety violations and performance metrics are affected by noise levels across various maneuvers and attack durations. Among all the maneuvers, straight-line driving (10m attack) demonstrates the highest correlation between noise levels and mission failure, with a coefficient of 0.99, indicating that shorter attack duration in straight-line driving are highly sensitive to noise. The overtake scenario follows this with a correlation of 0.93. Both the intersection and straight-line 20m scenarios show a correlation of 0.84 for mission failure, suggesting that longer attack duration and intersection maneuvers are somewhat less sensitive to noise, possibly due to the nature of the mission. Regarding sidewalk incursions, longer attack duration in the straight-line (20m) and overtake scenarios show the strongest correlations, at 0.98 and 0.96, respectively. In contrast, the intersection maneuver displays the weakest correlation for sidewalk incursions, reflecting the controlled, slower nature of this maneuver.

When examining localization loss, straight-line 10m and overtake show the highest correlations, 0.97 and 0.92, respectively, indicating that these scenarios are most affected by noise in terms of localization. The intersection scenario, though still sensitive to noise (0.86), shows a somewhat lower correlation, likely due to the AV's reduced speed and static behavior at stop points. Collision, on the other hand, shows similarly strong correlations in overtaking (0.84) and intersection (0.87) scenarios, but this metric is irrelevant in straight-line driving, as there are no NPCs involved in those maneuvers. The correlation for RollOut switches is also highest in straight-line 10m attacks (0.97), followed by straight-line 20m and overtake, while intersections have the lowest correlation (0.71) in this category. For NDTer, longer attack durations in straight-line scenarios show the highest correlation (0.92), while intersections and overtakes show lower values.

Overall, the straight-line (10m) and overtake scenarios exhibit the highest sensitivity to noise across several metrics, such as mission failure, sidewalk incursions, and localization loss. Intersection scenarios, in contrast, show consistently lower correlations, likely due to the nature of the maneuver, where the vehicle slows down or stops, reducing the dynamic impact of noise during attacks. This behavior at intersections explains the weaker overall correlation with noise, as the AV is generally at lower speeds and is less engaged in continuous movement compared to the overtake and straight-line scenarios. This highlights how the nature of each maneuver, particularly its dynamic or static characteristics, influences the vehicle's vulnerability to noise-induced safety violations and performance degradation.

VIII. Discussion

Throughout the paper, we demonstrated that AD software is sensitive to EMI attacks that can generate different levels of safety violations from low-priority violations, from which the vehicle can recover but resulting in suboptimal behaviour, to severe violations causing collisions or endangering other road users.

RQ1 How does a manipulation to the electromechanical component propagate through the AD software stack?

From our results, it emerges that an EMI attack at the steering sensor level often causes SiIn, DTL, or DTC violations, which are the most commonly visible in Fig. 8. To back-step this behaviour, to eventually debug such a complex AD software stack in a general purpose approach, developers will require an accurate analysis of each block in terms of data input-output relation. In our case, we carried out a back-step analysis at the ROS-topic level to identify the nodes that subscribe to specific messages. Here, we found out that the most probable user of steering sensor data, thus generating violations, is the mission and motion planning module, visible in Fig. 4, and composed of several sub-blocks including `op_trajectory_generator` and `op_waypoint_follower`, that represent the most probable components generating wrong decisions. While at the low level, PID controllers might be able to withstand noise to some extent, intelligent controllers have shown inherent vulnerability to this attack propagating from the low level up as raw sensor data to the master controller and up to the ROS topics.

RQ2 What dependencies exist between the AD control algorithm and low-level control?

High-level intelligent controllers trust digital data flowing over the in-vehicle network communication level. The interdependence of control algorithms resides in the feedback loop reading data from the low level while the AD acts in a hybrid deliberate/reactive robotic paradigm. In such a paradigm, well studied in robotics, an AD reacts quickly upon sensing without performing global-planning, which is typically a computationally demanding task running concurrently. SiIn, DTL, or DTC violations, which are the most commonly found in our analysis, are a typical result of the reactive behaviour of ADs. Similarly, the planner might generate unsafe trajectories in case of localization data corruption such as NDTLs violation or increase in NDTer margin. Eventually, the vehicle can recover from some violation when the global-planner generates a new waypoint, but this is not always guaranteed when some stochasticity is involved in the process.

RQ3 Where in the architecture of the autonomous vehicle can defensive mechanisms be placed to defend against control invariants?

Strategies to detect and mitigate low-level sensor data input manipulation focus on redundancy and multiple levels of data integrity checks. To investigate this question we step through each of the layers of the AV:

- **Low-Level PID Controller:** Integrity and plausibility checking of the PID can mitigate but not stop the injection of anomalous sensor input values. The PID has its own robustness, which is mathematically proved, the PID lacks the intelligence to interpret the meaning behind the input data. Therefore, attacks which manipulate the sensor input always have the possibility of traversing the PID. It is also possible to implement analog filters and hardware saturation, however, as mentioned, at this level, there is no means to discern attack behaviour which resembles regular signal/circuit specification and its operating characteristic.
- **Intermediate Layer:** At this level, it is possible to conduct inspection of the CAN data. The master controller has low-computational capacity. Therefore, implementation of mechanisms to interpret and provide intelligence of the CAN data is limited. Data saturation and filtering is possible at this level. However, filtering and saturation strategies would be challenged to defend against an adaptive sensor manipulation attack which searches for the filtering and saturation parameters and develop a 1-step or n-step attack which falls outside the range.
- **High-Level Control Layer:** A redundant, fall-back controller has a cost in terms of financial, compute and network resources, and cannot guarantee that an attack would also aim to manipulate the redundant controller. Furthermore, redundant controllers accessing the same sensor data might generate the same unexpected behaviour.

Our recommendations, for this particular use case, is to accurately model the sensor behaviour at the physical level considering the physical world world we live in. In this context, sensors, such as everything else, should obey Newton (for motion) and Maxwell equations (for electromagnetism). To detect sensor data anomaly our knowledge of the physical model of the sensor can be utilised to predict variances to this model. This would effectively detect a possible attack much earlier and thus prevent DTC & DTL violations occurring in the motion planning block. The validation of sensor data can run in a concurrent process throwing exceptions in case of unexpected levels of noise. The response action to an exception need to be modelled on the level of risk.

IX. Related Work

The closest work to our study is that of Berdich and Groza [21], which conducted multiple injection attacks (Fuzzing, Replay, DDoS) within the CAN Bus, targeted at diverse low-level sensors (Steering, Braking, Advanced Driver Assistance Systems (ADAS) ECU), within a cruise-control vehicle architecture. The experiment is only conducted in Simulink and the wider software stack including control properties and high-fidelity sensing of the vehicle are not included. The experimentation is conducted on the basis of establishing the feasibility of attack and model potential risk and safety consequences. Similarly, Pöllny et al. [6] developed an EMI attack using a helmholz coil which successfully manipulated a sensor popular used in ADAS. The study which focused at the low-level recommenced the possibility of plausibility check to mitigate steering angle attacks. Within, automotive software, studies of Garcia et al. [22] and Kim et al. [19] have discussed the problem of attacks on the low-level control with software developers, discovering issues with software implementation and development of defensive mechanisms, however, the scope of these studies did not include practical experimentation.

X. Conclusion

Our study analysed the robustness of a real-world AV to attacks on the low-level actuation using the example of an EMI attack on the steering angle sensor. We developed a hybrid low-level sensor and high-fidelity simulation environment, which we used to conduct approx. 1900 runs of diverse attack and driving scenarios. Our results demonstrated that our real-world AV is vulnerable to attacks on the low-level actuation. The effects of these attacks demonstrate that manipulated sensor input can propagate through to the higher-level control, and, in our case, impact modules for AD such as localisation and planning. Given the reactive nature of the AD to sensing, once a malicious input is within the system, the vehicle is to shown to react to this input with unsafe driving actions. Our recommendation to designers of AV software is to accurately model the sensor behaviour at a physical level and use this knowledge to predict variances in the model. This would provide the ability to detect a possible attack earlier and prevent collisions and localisation loss.

Acknowledgment

This work is supported by the Horizon Europe project CyberSecDome (Agreement No.: 101120779) and EU Horizon 2020 project MariCybERA (Agreement No.: 952360) and by the European Commission through the H2020 teaming project Finest Twins (Grant No. 856602).

References

[1] F. Zhang, H. A. D. E. Kodituwakku, J. W. Hines, and J. Coble, "Multilayer data-driven cyber-attack detection system for industrial control systems based on network, system, and process data," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4362–4369, 2019.

[2] C.-V. Briciu, I. Filip, and F. Heininger, "A new trend in automotive software: Autosar concept," in *2013 IEEE 8th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, 2013, pp. 251–256.

[3] F. Munir, S. Azam, M. I. Hussain, A. M. Sheri, and M. Jeon, "Autonomous vehicle: The architecture aspect of self driving car," in *Proceedings of the 2018 International Conference on Sensors, Signal and Image Processing*, ser. SSIP '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1–5. [Online]. Available: https://doi.org/10.1145/3290589.3290599

[4] N. United States National Transportation Safety Board, "Investigation of lion air flight 610 and ethiopian airlines flight 302," *Safety Recommendation Report NTSB ASR1901. PDF document*, 2019.

[5] E. S. I. B. ESIB, "Accident, loss of control with airbus a320-214 near tallinn airport on 28.02.2018." *Safety Investigations. Investigation report ESIB: A2802118 EECAIRS: EE0180. PDF document.*, 2019.

[6] O. Pöllny, F. Kargl, and A. Held, "Steering your car with electromagnetic fields," in *Proceedings of the 6th ACM Computer Science in Cars Symposium*, ser. CSCS '22. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: https://doi.org/10.1145/3568160.3570228

[7] Y. Tu, V. S. Tida, Z. Pan, and X. Hei, "Transduction shield: A low-complexity method to detect and correct the effects of emi injection attacks on sensors," in *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, ser. ASIA CCS '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 901–915. [Online]. Available: https://doi.org/10.1145/3433210.3453097

[8] Y. Zhang and K. Rasmussen, "Detection of electromagnetic interference attacks on sensor systems," in *2020 IEEE Symposium on Security and Privacy (SP)*, 2020, pp. 203–216.

[9] Y. Tu, Z. Lin, I. Lee, and X. Hei, "Injected and delivered: Fabricating implicit control over actuation systems by spoofing inertial sensors," in *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, Aug. 2018, pp. 1545–1562. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/tu

[10] T. Trippel, O. Weisse, W. Xu, P. Honeyman, and K. Fu, "Walnut: Waging doubt on the integrity of mems accelerometers with acoustic injection attacks," in *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, 2017, pp. 3–18.

[11] P. Dash, M. Karimibiuki, and K. Pattabiraman, "Stealthy attacks against robotic vehicles protected by control-based intrusion detection techniques," *Digital Threats*, vol. 2, no. 1, jan 2021. [Online]. Available: https://doi.org/10.1145/3419474

[12] S. Jha, S. Banerjee, T. Tsai, S. S. Hari, M. B. Sullivan, Z. T. Kalbarczyk, S. W. Keckler, and R. K. Iyer, "Ml-based fault injection for autonomous vehicles: A case for bayesian fault injection," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. Los Alamitos, CA, USA: IEEE Computer Society, jun 2019, pp. 112–124. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/DSN.2019.00025

[13] L. J. Moukahal, M. Zulkernine, and M. Soukup, "Boosting grey-box fuzzing for connected autonomous vehicle systems," in *2021 IEEE 21st International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, 2021, pp. 516–527.

[14] H. Choi, W.-C. Lee, Y. Aafer, F. Fei, Z. Tu, X. Zhang, D. Xu, and X. Deng, "Detecting attacks against robotic vehicles: A control invariant approach," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 801–816. [Online]. Available: https://doi.org/10.1145/3243734.3243752

[15] H. J. Jo and W. Choi, "A survey of attacks on controller area networks and corresponding countermeasures," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 6123–6141, 2022.

[16] A. Buscemi, I. Turcanu, G. Castignani, A. Panchenko, T. Engel, and K. G. Shin, "A survey on controller area network reverse engineering," *IEEE Communications Surveys and Tutorials*, vol. 25, no. 3, pp. 1445–1481, 2023.

[17] S. K. T. U. of Tokyo), "Autoware: Ros-based oss for urban self-driving mobility," in *ROSCon Vancouver 2017*. Open Robotics, September 2017. [Online]. Available: https://doi.org/10.36288/ROSCon2017-900813

[18] H. Darweesh, E. Takeuchi, and K. Takeda, "Openplanner 2.0: The portable open source planner for autonomous driving applications," in *2021 IEEE Intelligent Vehicles Symposium Workshops (IV Workshops)*, 2021, pp. 313–318.

[19] H. Kim, R. Bandyopadhyay, M. Ozmen, Z. Celik, A. Bianchi, Y. Kim, and D. Xu, "A systematic study of physical sensor attack hardness," in *2024 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, may 2024, pp. 146–146. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SP54263.2024.00143

[20] G. Lou, Y. Deng, X. Zheng, M. Zhang, and T. Zhang, "Testing of autonomous driving systems: where are we and where should we go?" in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2022. New York, NY, USA: Association for Computing Machinery, 2022, p. 31–43. [Online]. Available: https://doi.org/10.1145/3540250.3549111

[21] A. Berdich and B. Groza, "Cyberattacks on adaptive cruise controls and emergency braking systems: Adversary models, impact assessment, and countermeasures," *IEEE Transactions on Reliability*, vol. 73, no. 2, pp. 1216–1230, 2024.

[22] J. Garcia, Y. Feng, J. Shen, S. Almanee, Y. Xia, Chen, and Q. Alfred, "A comprehensive study of autonomous vehicle bugs," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, ser. ICSE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 385–396. [Online]. Available: https://doi.org/10.1145/3377811.3380397

Appendix V

Paper V

M. Hamad, A. Finkenzeller, M. Kühr, **A. Roberts**, O. Maennel, V. Prevelakis, and S. Steinhorst. React: Autonomous intrusion response system for intelligent vehicles. Computers & Security, 145:104008, 2024.

Computers & Security 145 (2024) 104008

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

REACT: Autonomous intrusion response system for intelligent vehicles

Mohammad Hamad [a,*], Andreas Finkenzeller [a], Michael Kühr [a], Andrew Roberts [c], Olaf Maennel [d], Vassilis Prevelakis [b], Sebastian Steinhorst [a]

[a] *Technical University of Munich, Munich, Germany*
[b] *Technical University of Braunschweig, Braunschweig, Germany*
[c] *Tallinn University of Technology, Tallinn, Estonia*
[d] *University of Adelaide, Adelaide, Australia*

ARTICLE INFO

Keywords:
Security
Intrusion response system
Intelligent vehicle

ABSTRACT

Autonomous and connected vehicles are rapidly evolving, integrating numerous technologies and software. This progress, however, has made them appealing targets for cybersecurity attacks. As the risk of cyber threats escalates with this advancement, the focus is shifting from solely preventing these attacks to also mitigating their impact. Current solutions rely on vehicle security operation centers, where attack information is analyzed before deciding on a response strategy. However, this process can be time-consuming and faces scalability challenges, along with other issues stemming from vehicle connectivity. This paper proposes a dynamic intrusion response system integrated within the vehicle. This system enables the vehicle to respond to a variety of incidents almost instantly, thereby reducing the need for interaction with the vehicle security operation center. The system offers a comprehensive list of potential responses, a methodology for response evaluation, and various response selection methods. The proposed solution was implemented on an embedded platform. Two distinct cyberattack use cases served as the basis for evaluating the system. The evaluation highlights the system's adaptability, its ability to respond swiftly, its minimal memory footprint, and its capacity for dynamic system parameter adjustments. The proposed solution underscores the necessity and feasibility of incorporating dynamic response mechanisms in smart vehicles. This is a crucial factor in ensuring the safety and resilience of future smart mobility.

1. Introduction

In recent years, there has been remarkable progress in the development of smart vehicles. Today's vehicles resemble interconnected networks on wheels, with numerous embedded computers, called Electronic Control Units (ECUs), linked through various types of networks, hosting an extensive number of software components totaling over a hundred million lines of code. Moreover, these networks incorporate various intelligent sensors (such as cameras, LiDAR, radar, etc.) and different connectivity technologies that enhance the vehicle's ability to perceive and interact with the surrounding environment, thus bolstering autonomy and minimizing the reliance on human intervention. However, with the rise of connectivity and the softwarization of vehicles, the vulnerability to cyberattacks targeting these systems has also escalated (Upstream, 2022).

Recently, there has been a growing interest in addressing the security threats that may target smart vehicles. For instance, the ISO 21434 (International Organization for Standardization, 2021) standard has been introduced, with a significant portion dedicated to the development of threat analysis and risk assessment methodologies. Moreover, the field of intrusion detection and prevention in the automotive domain has witnessed extensive research, leading to various avenues for research (Kim et al., 2021). However, despite these efforts, the number of attacks targeting smart vehicles continues to rise (Upstream, 2022). This is to be expected, as security is not absolute, and we must acknowledge that complete prevention of all security threats may not be attainable. Therefore, greater emphasis should be placed on defining *how the system should behave when confronted with such unavoidable attacks.*

The cybersecurity incident response is an integral aspect of security management, as outlined in ISO/SAE 21434 within the operational and maintenance clause (International Organization for Standardization, 2021). Based on the standard, this process aims to provide remedial

* Corresponding author.
E-mail addresses: mohammad.hamad@tum.de (M. Hamad), andreas.finkenzeller@tum.de (A. Finkenzeller), michael.kuehr@tum.de (M. Kühr), andrew.Roberts@taltech.ee (A. Roberts), olaf.maennel@adelaide.edu.au (O. Maennel), prevelakis@ida.ing.tu-bs.de (V. Prevelakis), sebastian.steinhorst@tum.de (S. Steinhorst).

https://doi.org/10.1016/j.cose.2024.104008
Received 15 January 2024; Received in revised form 5 June 2024; Accepted 19 July 2024
Available online 23 July 2024
0167-4048/© 2024 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

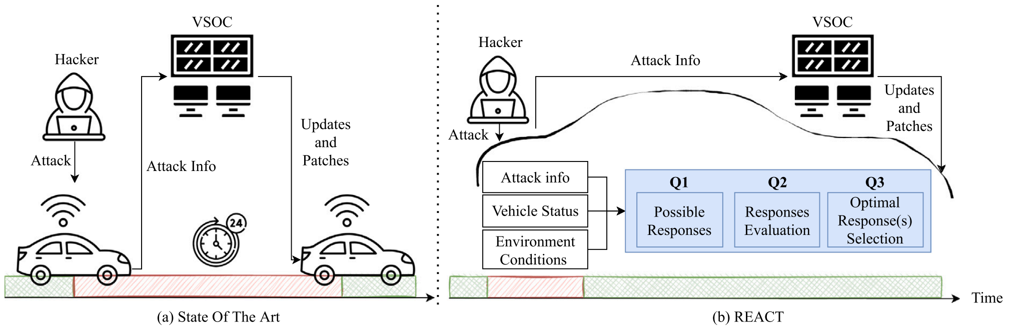


Fig. 1. On the left side, the current vehicle system shares attack information with the VSOC but often has to wait for extended periods to receive necessary security patches and updates. This waiting period puts the vehicle in a malicious status (red, diagonal lines). On the right side, the vehicle can select and implement security solutions to avoid the long waiting time for security patches and updates and return to normal status (green, cross diagonal lines).

actions and updates, which may involve post-development changes to address security vulnerabilities. The process necessitates the vehicle to share cybersecurity information about the vulnerability that triggered the cybersecurity incident response. Being part of the ISO/SAE 21434, it is now imperative that manufacturers comply with new regulations by having a cybersecurity management system that oversees the cybersecurity activities and processes in the product life-cycle. To achieve this, Vehicle Security Operation Centers (VSOCs) will be utilized to support monitoring (Barletta et al., 2023; Sembera, 2020; Olt, 2019). Such VSOCs will employ expert teams that continuously analyze data collected from all connected vehicles, enabling automakers to swiftly and efficiently address security incidents (Olt, 2019). Although it is arguable that numerous tasks within a VSOC could be automated, the challenge of scalability persists, especially considering the extensive fleet of connected vehicles and the immense data volumes accumulated by each vehicle, reaching terabytes (Wright, 2021). The transfer and processing of such data turn out to be significant issues, particularly in urban areas with hundreds of cars per vicinity, leading to bottlenecks. Additionally, the connectivity itself could be an attractive target for attackers. In this context, the integration of VSOCs into the smart vehicle ecosystem demands solutions for addressing connectivity challenges between vehicles and the VSOC, as well as managing privacy concerns tied to shared data (Hamad and Steinhorst, 2023).

Finally, and more importantly, there is a need to ensure a near-real-time response to security attacks. Taking into account the need for a human in the loop, as well as the latency introduced by high-volume shared data and communication between the vehicles and the VSOC, achieving a near-real-time response seems unrealistic. This perspective is supported by the European Union Agency for Cybersecurity (ENISA), which has cautioned that responding to high-criticality attacks could potentially take days or even weeks (ENISA, 2019). The scenario of extended waiting presents a dilemma, with two options, each having its own disadvantages. Allowing a vehicle to operate with a compromised component due to extended waiting for a security update is far from the ideal situation. Alternatively, suspending the compromised component until the security update is received might not be the best course of action either, particularly if the component plays a crucial role in operations.

Contributions: Therefore, there is a need for vehicles to be equipped with the capability to swiftly respond to cyberattacks. However, having such a capability requires the answering of three main questions (see Fig. 1): **Q1:** What are the possible responses that can be taken? **Q2:** What factors need to be considered when evaluating these responses? **Q3:** How to select one or more of these responses at the run-time based on the responses' evaluation? This paper aims to address these questions by investigating and categorizing potential responses according to the impact of various cyber attacks to which each response aims to react. Additionally, the paper presents a dynamic risk assessment and cost evaluation for attacks and responses, utilizing given data such as attack information and vehicle status. This assessment supports the selection of suitable responses. Furthermore, the paper explores different approaches for response selection, conducts comparisons, and identifies those best suited for automotive systems. Lastly, the paper introduces an intrusion response system, referred to as REACT, evaluates it using two attack scenarios, and discusses both the quality of the responses it generates and its overall efficiency. In summary, the main contributions of this paper are as follows:

- We conduct a comprehensive review of existing intrusion response strategies for IT systems and map them to automotive systems, considering the unique characteristics of automotive attacks and automotive system architectures (see Section 2).
- We propose a novel method for calculating the cost and response benefits by extending existing risk assessment approaches specific to automotive systems (see Section 3).
- We explore a range of algorithms for selecting appropriate responses, conduct comparative analyses, and identify the most suitable algorithms for automotive systems, proposing their adoption to enhance automotive security (see Section 4).
- We introduce REACT, a comprehensive automotive IRS, and provide an open-source prototype[1] (see Section 5).
- We demonstrate the feasibility and applicability of the proposed automotive IRS through evaluations using embedded platforms and two attack scenarios. Findings indicate that the system can adapt to different scenarios, makes response selections quickly (average 30 ms for the worst-case algorithm), has low memory overhead, and dynamically adjusts system parameters (see Section 6).

2. Response strategies

The purpose of this section is to address the first question (**Q1**) about possible response strategies. To do so, it is critical to have a deep understanding of the system as well as the potential attacks and threats it may face. Therefore, this section introduces the design of an automotive reference architecture, discusses the potential threats that may arise, and provides a comprehensive summary of the different response strategies that can be utilized to mitigate these attacks.

[1] https://github.com/mohammadhamad/REACT.

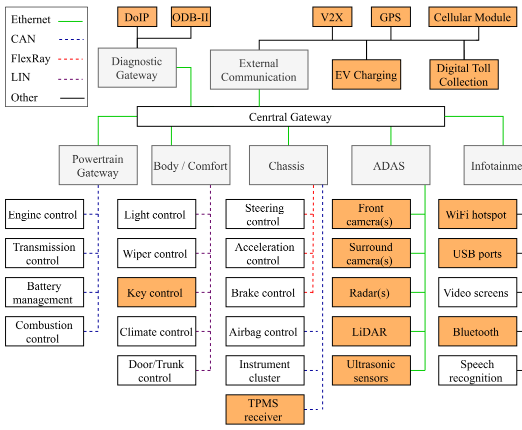


Fig. 2. Reference vehicle architecture with possible attack surfaces (orange).

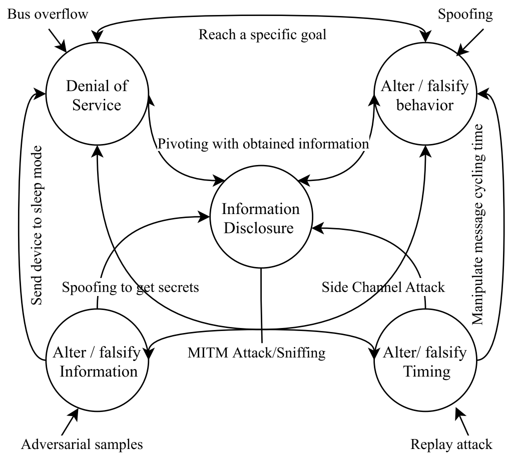


Fig. 3. Classification of intrusion results and examples of attacks for each possible intrusion result.

2.1. Automotive reference architecture

In order to understand how IRS can be integrated into modern vehicles and the potential responses they can provide, it is essential to first understand their system architecture. Fig. 2 presents a generic, realistic and comprehensive reference architecture that can be found in modern vehicles. It is notable that a modern vehicle includes *highly interconnected* subsystems. The figure also shows how modern vehicles have many *embedded devices*, known as ECUs, which are *distributed* allover the vehicle, communicating among themselves via different types of networks such as CAN, Flexray and Ethernet. These ECUs are grouped in different domains or zones based on the functionality such as infotainment, Advanced Driver Assistance Systems (ADAS), powertrains, etc. Besides ECUs, modern vehicles are equipped with many sensors (e.g., cameras, LiDAR, etc.), advanced communication technology for connecting with the external world, and diagnostic ports (e.g., OBD-II) that collectively form a significant attack surface for different types of attacks and threats (Checkoway et al., 2011). The unrestricted or/and uncontrolled interaction among all those components puts the whole system in danger. Attackers could launch a *stepping-stone* attack (Ullah et al., 2020), where they compromise a non-critical ECU with weaker security (e.g., the infotainment system), in order to gain control of a more crucial one (e.g., engine control) (Miller and Valasek, 2015; Costantino and Matteucci, 2023). All these characteristics of the vehicle architecture suggest that any proposed IRS should take into account the constrained resources and the highly interconnected and distributed nature of a vehicular system.

2.2. Threats and attacks

Threat Analysis and Risk Assessment (TARA), an essential component of ISO 21434, is employed as a systematic way to identify and assess cybersecurity threats and risks in the automotive industry, facilitating the implementation of effective mitigation strategies. Since TARA does not dictate a specific method to identify threats, various methods have been proposed, such as STRIDE (Karahasanovic et al., 2017), SAVTA (Hamad and Prevelakis, 2020), attack trees (Henniger et al., 2009; Hamad et al., 2016), and many others (Luo et al., 2021). Following the methodology of TARA, these methods provide a comprehensive list of threats and attacks that may target the vehicular system and offer preventive measures. However, they do not address the reactive measures required for an automotive IRS.

Using the list of threats and attacks to create a response for each of them seems to be not ideal due to several challenges, including the large number of attacks and the requirements for precise information about each attack, which must be provided by the Intrusion Detection System (IDS). This challenge becomes evident when considering Zero-Day attacks, where information about such attacks may not be available to the IRS at the time of detection by the IDS. Even if an anomaly-based IDS shares some information about the attack pattern with the IRS, a response solely based on known attack patterns may not sufficiently react to these Zero-Day attacks. Therefore, the most effective approach is to enable the IRS to understand the situation it aims to respond to. This involves focusing on the impact or outcome of different attacks rather than solely on the attacks themselves.

To achieve that, we have developed a model, illustrated in Fig. 3, which represents the actual results of intrusions collected from various research works. The model encompasses five main attack outcomes, each of which can result from multiple types of attacks. Examples of these attacks are depicted in the outer nodes of Fig. 3. Also, to reflect the outcome of stepping-stone attacks, the model links the different outcomes to demonstrate that certain attacks may cause a series of results. The five attack outcomes are:

- *Falsify/Alter Information:* Different attacks have the potential to modify information on a bus or within an ECU. It is important to note that not every alteration of information automatically results in undesirable behavior. For instance, adversarial samples (Mahima et al., 2021), such as incorrect classifications of objects detected by a camera, may not necessarily lead to incorrect behaviors.
- *Falsify/Alter Timing:* This outcome typically occurs as a result of attacks targeting the communication buses of the vehicle (Wolf et al., 2004; Lokman et al., 2019) or the real-time tasks on the ECUs (Hamad et al., 2018).
- *Information Disclosure:* This outcome is the result of attacks, such as spoofing, eavesdropping, and others, that aim to allow attackers to gain unauthorized access to sensitive information exchanged during communication or stored within the ECUs (Cui et al., 2019).
- *System Unavailability:* This outcome typically occurs as a result of Denial of Service (DoS) attacks that aim to cause a loss of availability for a specific component or subsystem in the vehicle (Palanca et al., 2017). Such attacks can lead to severe damage to the system, especially if they target high-critical components (Alrefaei et al., 2022).

Table 1
Classification of generic responses to intrusion results.

Intrusion result	Response index. response
Falsify/Alter timing	**1**. Use of redundant information (Hamad et al., 2021), **2**. Correction of timing (Papadaki et al., 2003; El-Rewini et al., 2020), **3**. Force additional authentication (Anwar et al., 2015), **4**. Restart the device/system (Kholidy et al., 2016), **5**. Change settings (Hughes et al., 2020), **6**. Redirect traffic (Hughes et al., 2020), **7**. Re-initialization (Herold, 2017)
Falsify/Alter information	**1**. Use of redundant information (Reallocation) (Hamad et al., 2021), **3**. Force additional authentication (Anwar et al., 2015), **4**. Restart the device/system (Kholidy et al., 2016), **8**. Create a backup (Chevalier et al., 2019), **5**. Change settings (Hughes et al., 2020), **7**. Re-initialization (Herold, 2017), **9**. Correct protocol specification faults (Herold et al., 2016), **10**. Split or merge functions (Yarygina and Otterstad, 2018)
Information disclosure	**11**. Issue authentication challenges (Papadaki et al., 2003), **12**. Re-enforce access control (Anuar et al., 2012), **3**. Force additional authentication (Anwar et al., 2015), **13**. Introduce a honeypot (Anuar et al., 2012), **4**. Restart the device/system (Kholidy et al., 2016), **14**. Modify firewall (Hughes et al., 2020), **6**. Redirect traffic (Hughes et al., 2020), **10**. Split or merge functions (Yarygina and Otterstad, 2018), **7**. Re-initialization (Herold, 2017), **15**. Network isolation (El-Rewini et al., 2020)
System unavailability	**1**. Use of redundant information (Reallocation) (Hamad et al., 2021), **12**. Re-enforce access control (Anuar et al., 2012), **13**. Introduce a honeypot (Anuar et al., 2012), **4**. Restart the device/system (source or destination) (Kholidy et al., 2016), **14**. Modify firewall (Hughes et al., 2020), **6**. Redirect traffic (Hughes et al., 2020), **10**. Split or merge functions (Yarygina and Otterstad, 2018), **7**. Re-initialization (Herold, 2017), **16**. Limit resources of the attacker (Chevalier et al., 2019), **17**. Safe mode (Hamad et al., 2019)
Falsify/Alter behavior	**1**. Use of redundant information (Reallocation) (Hamad et al., 2021), **18**. Correction of behavior (Papadaki et al., 2003), **9**. Correct protocol specification faults (Herold et al., 2016), **3**. Force additional authentication (Anwar et al., 2015), **19**. Restart the miss-behaving system (Kholidy et al., 2016), **5**. Change settings (Hughes et al., 2020), **10**. Split or merge functions (Yarygina and Otterstad, 2018), **7**. Re-initialization of the miss-behaving device (Herold, 2017), **17**. Safe mode (Hamad et al., 2019), **8**. Create a backup (Chevalier et al., 2019)
General	**20**. Isolation (Hamad et al., 2021), **21**. Limit communication of malicious system (Hamad et al., 2021), **22**. Drop packets (Kholidy et al., 2016), **23**. Trace communication (Hamad et al., 2021), **24**. Introduce additional logging (Anwar et al., 2015), **25**. Block network traffic (Anuar et al., 2012), **26**. Kill process (Hamad et al., 2021), **27**. Reduce trust level of the source (Hamad et al., 2021), **28**. Perform a security auditing (Hamad et al., 2019), **29**. Request/Perform software update (Papadaki et al., 2003), **30**. Notify Security Operations Center (SOC)/administrator (Anwar et al., 2017; Anuar et al., 2012), **31**. No action (Anwar et al., 2017), **32**. Adapt parameters for IDS (Heigl et al., 2018), **33**. Warn/inform other ECUs (AUTOSAR, 2020; Hamad et al., 2021)

- *Falsify/Alter behavior:* This outcome is the result of tampering attacks that specifically target the components, data, or parameters of a system with the intention of altering the system's intended behavior and achieving unauthorized or malicious outcomes (Miller and Valasek, 2015). While this intrusion outcome may appear similar to falsify/alter information, the key distinction is that in falsify/alter information attacks, the goal is to tamper with the information itself without the explicit method of changing the system's behavior, even though it may indirectly lead to such changes.

2.3. Response possibilities

After classifying the outcome of the attack, it becomes easier to determine which responses can be used to address that particular outcome and handle the attacks that cause it. In order to do so, we have examined typical responses discussed in both the automotive and non-automotive domains. It should be noted that while some research papers in the automotive domain have discussed the need for responses to certain attacks, there is currently no comprehensive research that lists and classifies all possible responses. Furthermore, it is important to consider that some of the responses we collected were originally designed for computer networks and may not be directly applicable to automotive bus systems due to the lack of specific security mechanisms (El-Rewini et al., 2020). For example, response actions such as IP address changes or port blocking (Anwar et al., 2015) are highly specific to Ethernet and higher protocols such as IP, and therefore have limited suitability for certain aspects of communication in vehicles. To address this challenge, we have defined a list of generic responses that are specific enough to be applied in an automotive IRS, while also being adaptable to constrained and potentially insecure devices. Table 1 provides an overview of the different responses based on the identified attack outcomes. In addition, we have included a "General" category that encompasses responses applicable to all five categories. For more detailed information about each response, please refer to the respective sources cited in Table 1.

3. Dynamic cost and impact evaluation

In this section, we will address **Q2** by outlining the key factors required to enable the selection of the most effective response by the IRS. These factors can be categorized into two groups: *intrusion-related factors*, which pertain to the attack's impact and risk, and *response-related factors*, which concern the cost and benefit of the chosen response.

3.1. Intrusion-related factors

3.1.1. Intrusion properties

For each detected intrusion, the following properties need to be determined:

- *Source of the intrusion:* This represents the component from which the attack was launched. Referring to the automotive reference architecture depicted in Fig. 2, sources can include entities from the attack surface as well as external attackers targeting any of these components.
- *Destination of the intrusion:* The attacked entity can be described as the destination of the intrusion. This could be ECUs, sensors, or bus systems.
- *Intrusion result:* This refers to one of the outcomes that were previously defined in Section 2.2. Similar to the source and destination of an intrusion, this information is also provided by an IDS.
- *Intrusion impact*: This information serves to depict the impact of the intrusion on the system and is essential for evaluating the risks during the attack.

3.1.2. Dynamic attack impact assessment

To assess the potential risks associated with an intrusion, it is necessary to understand the impact of the attack and the likelihood of its occurrence (International Organization for Standardization, 2021; Lautenbach et al., 2021). To calculate the impact of the intrusion, many methods were already adopted such as HEAVENS (Islam et al., 2016). HEAVENS classifies the impact of a given threat based on four metrics (Wang et al., 2021b; Luo et al., 2021):

1. Safety impact, denoted as S with $S \in \{0, 10, 100, 1000\}$
2. Financial impact, denoted as F with $F \in \{0, 10, 100, 1000\}$
3. Operational impact, denoted as O with $O \in \{0, 1, 10, 100\}$
4. Privacy impact, denoted as P with $P \in \{0, 1, 10, 100\}$

In the original HEAVENS method, the overall impact I is calculated as a sum of the four single impacts as depicted in Eq. (1) (Wang et al., 2021b).

$$I = S + F + O + P \tag{1}$$

One issue with the impact calculation, as presented in Eq. (1), is the overemphasis on safety and financial parameters. This skewed emphasis not only complicates the comparison and independent evaluation of the four metrics but also renders it unsuitable for an automotive IRS. In the automotive context, safety and operational considerations typically outweigh financial and privacy-related aspects for most automotive functions. Considering the aforementioned issue, we propose normalizing all possible values to $0, 1, 10, 100$, representing no, low, medium, or high impact for each of the four metrics in HEAVENS.

Another limitation of the current risk assessment methods, including HEAVENS, is their failure to account for dynamic environmental factors, such as run-time context, operational status, and the surrounding environment. This gap may arise because HEAVENS is primarily applied during the design phase, making it somewhat oblivious to runtime conditions. To address this challenge and enhance the method's applicability for use within automotive IRS, we introduce a new metric termed "Environment", denoted as E. This metric, E, encompasses dynamic factors that are crucial for assessing intrusion impact (Hamad et al., 2021). Potential inputs that can be used to derive the environmental parameter E include vehicle speed, road conditions, the proximity of nearby objects, and more. These parameters can exert significant influence, as a single intrusion may yield different impacts depending on physical and environmental considerations.

The final enhancement option for the HEAVENS method involves the capability to dynamically adjust the assessment of intrusion impact. Following a successful intrusion response, it may become evident that the stored parameters for S, F, O, P, and E require a different representation. HEAVENS currently confines impact values to $0, 1, 10, 100$, and a simple adjustment to a new value could result in significant overrepresentation. To address this issue, introducing weights for each of the five evaluation metrics (w_S, w_F, w_O, w_P, and w_E) offers a valuable mechanism for accommodating learning and adaptation processes. The optimization proposals discussed earlier to transform the calculation of intrusion impact using the HEAVENS method into a dynamic process lead to Eq. (2).

$$I = w_S \cdot S + w_F \cdot F + w_O \cdot O + w_P \cdot P + w_E \cdot E \tag{2}$$

Utilizing dynamically adjusted static values for S, F, O, and P, each incorporating their respective weights, in addition to dynamically acquired values for E along with an adapted static weight. In cases involving specific automotive architectures, the equation can also be applied in a more granular fashion for particular assets. Initial values for all these parameters can be established by security experts, drawing upon their experiential knowledge.

The source and destination of the attack are employed to determine the attack's location, aiding in the calculation of the subsequent attack likelihood, especially when considering step-stone attacks, across various parts of the system. This assessment of attack likelihood, in conjunction with the evaluation of attack impact, contributes to the overall risk assessment.

3.2. Response-related factors

3.2.1. Response properties

Similar to the intrusion, each response will have five properties that need to be identified:

- *Actual action:* They refer to the actual actions taken in the event of an intrusion. These actions can be selected from those presented in Table 1.
- *Precondition:* Some responses may require preconditions that must be met. These preconditions can be expressed as Boolean expressions and serve as prerequisites to trigger the response.
- *Place of application:* Refers to the location where the response will be implemented. A response can be applied either at the source entity of an intrusion, the destination, or at both locations.
- *Stop condition:* Refers to the condition for which the implemented response should cease. This condition can be related to a specific time (Lopes and Hutchison, 2020), the successful reestablishment of security policies (Hamad et al., 2021), or the necessity for persistent measures (Ullah et al., 2020).
- *Cost and benefit of the response:* Refers to the costs and benefits incurred when implementing a response to an intrusion or security incident.

3.2.2. Dynamic response cost and benefit assessment

When considering the cost of responses, various methods were employed to determine their value in IT systems (Shameli-Sendi et al., 2012). These methods primarily rely on one of three models: a static cost model that assigns a fixed cost value for each response, a static evaluated cost model that calculates cost using a static function with some adjustment possibilities, or dynamic evaluated cost models that offer fully dynamic evaluation based on real-time data. Each model varies in terms of simplicity, adaptability, and accuracy, catering to different system requirements and scenarios.

Statically evaluated cost models provide a valid trade-off between achievable implementation efforts, especially on constrained devices similar to the ones used in automotive systems, and plausible results. These models maintain a static approach to calculating response costs, even though the actual cost values may vary. Various metrics for calculating response costs are mentioned in current literature. The first metric evaluates the impact of the response on availability (Shameli-Sendi et al., 2012). Availability's impact is represented as $A \in 0, 1, 10, 100$, with 0 meaning negligible and 100 meaning severe impact on availability, to ensure consistency with intrusion metrics. The second metric, describing the response cost, assesses its effect on the performance of the (sub)system (Shameli-Sendi et al., 2012), similar to the deployment cost of countermeasures (Guo et al., 2020). This metric is denoted as $Perf \in 0, 1, 10, 100$, with 0 meaning negligible impact on performance and 100 meaning severe impact on performance, to maintain a uniform scale with the impact of the response on availability.

To achieve results similar to the adapted HEAVENS method described in Section 3.1, a comparable equation can be employed to calculate the cost (c) of a response. By adopting specific weights (w_A and w_{Perf}) for the impact on availability and performance along with their actual values (A and $Perf$), the response cost can be computed as shown in Eq. (3). This approach results in a highly adaptable method for calculating the response cost. While the initial values for A and $Perf$ can be manually determined, they can also be adjusted over time. The specific weights offer a means to introduce a learning component within the mathematical framework.

$$c = w_A \cdot A + w_{Perf} \cdot Perf \tag{3}$$

Likewise, the adapted HEAVENS method introduced in Section 3.1 can be repurposed for evaluating the benefit of a response, with the exception of the environmental parameter E and its associated weight w_E. While HEAVENS assesses intrusion impact using four metrics, these same metrics can be employed to quantify the benefits in these four categories when assessing response value. By employing identical value possibilities with $S, F, O, P \in 0, 1, 10, 100$, a corresponding benefit value can be determined. The calculation of the benefit (b) for each response option, as shown in Eq. (4), is derived from Eq. (2).

$$b = w_S \cdot S + w_F \cdot F + w_O \cdot O + w_P \cdot P \tag{4}$$

Compared to existing research (Stakhanova et al., 2007; Guo et al., 2020), this repurposed HEAVENS method of Eq. (4) provides a more holistic approach on evaluating the benefit of applied responses. For each response option classified in Table 1, the cost calculated using Eq. (3) and the benefit determined using Eq. (4) must be applied, and preconditions must be established. Initial values for S, F, O, P, A, and $Perf$, along with their respective weights, can be assigned by security experts and subsequently updated either manually or through learning algorithms within an IRS. Similar to the impact calculation of intrusions, these weights can be adjusted to improve the accuracy of the model.

4. Optimal selection algorithms

In this section, we will address the third question **Q3**, by exploring numerous potential methods for selecting response strategies (Section 4.1), compare these approaches and provide a rationale for our chosen strategy (Section 4.2), and describe how to adopt the selected strategies (Section 4.3).

4.1. Possible algorithms

To determine the best method for selecting appropriate responses, we explore various algorithms and solutions used in *non-automotive domains* and compare them to identify the most suitable one that can be implemented within the vehicle system. Several surveys, such as Nespoli et al. (2018) and Bashendy et al. (2023a,b), provide valuable insights into response selection approaches in non-automotive domains, making them worth investigating for more comprehensive details.

4.1.1. SAW

SAW (Fishburn, 1967) is the simplest and most often used method. The basic concept of this method is to find a preference value (p) for each possible response, and then select the response with the highest preference value as the best option. To illustrate how this method works, let us assume that we have n possible responses ($\mathcal{R} = \{r_1, r_2, \dots, r_n\}$) and m criteria ($\mathcal{CR} = \{cr_1, cr_2, \dots, cr_m\}$) that will be used as a reference for evaluating the responses. Each criterion will be assigned a weight w_j where $\sum_{j=1}^{m} w_j = 1$. To calculate the preference values, a normalized decision matrix is first created, where each element of the matrix is normalized based on the nature of the criterion, whether it is a cost or benefit, as shown in Eq. (5).

$$\alpha_{ij} = \begin{cases} \frac{v_{i,j}}{\max_i(v_{i,j})}, & \text{if criterion } cr_j \text{ is a benefit} \\ \frac{\min_i(v_{i,j})}{v_{i,j}}, & \text{if criterion } cr_j \text{ is a cost} \end{cases} \tag{5}$$

where $v_{i,j}$ is the performance value of the response r_i when it is evaluated in terms of criterion cr_j. The preference value (p_i) of response r_i is then obtained by calculating the weighted sum of the normalized performance values using Eq. (6).

$$p_i = \sum_{j=1}^{m} w_j \cdot \alpha_{ij} \tag{6}$$

Finally, the response r_i with the highest preference value (p_i) is considered as the best selection response.

4.1.2. Linear Programming (LP)

LP is a mathematical technique that can be employed to select optimal responses (Herold et al., 2017). LP can be used to find the best combination of responses that *maximizes* or *minimizes* a certain objective function. To illustrate the workings of this method, let us consider a scenario where we have n possible responses ($\mathcal{R} = r_1, r_2, \dots, r_n$). The optimization of the objective function can be as in Eq. (7).

$$\sum_{i=1}^{n} x_i s_i \rightarrow \max \textit{ or } \min \tag{7}$$

where x_i represents a criterion related to the response r_i and $\vec{s}$ be a vector of binary decision variables, where s_i is equal to 1, it indicates that the corresponding response $r_i \in \mathcal{R}$ will be executed. Conversely, if s_i is equal to 0, it signifies that the response $r_i \in \mathcal{R}$ will not be executed. The optimization problem typically includes *constraints* to ensure the selection process adheres to specific conditions or limitations.

4.1.3. Game-theoretic algorithm

Another mathematical method to determine optimal responses against cyber attacks is game-theoretic algorithms (Yarygina and Otterstad, 2018; Zonouz et al., 2014; Wang et al., 2021a). In the game-theoretic approach, the attacker and the IRS are modeled as two players. Each player has a set of actions available to them, such as different attack strategies $\mathcal{A} = \{a_1, a_2, \dots, a_k\}$ for the attacker and response strategies $\mathcal{R} = \{r_1, r_2, \dots, r_n\}$ for the IRS. The goal of the IRS is to select the optimal response to the attack at a given time. One way to achieve that is by minimizing the maximum damage of the attack: $\min_{r_i \in \mathcal{R}}(\max_{a_i \in \mathcal{A}}(U(r_i, a_i)))$ where $U(r_i, a_i)$ represents the utility function for the IRS when the attacker chooses attack a_i and the IRS responds with response r_i.

4.1.4. AI-based mechanisms

Many AI-based mechanisms were used to support the dynamic selection of the response such as Genetic Algorithms (Fessi et al., 2009), Convolutional Neural Networks (Xia et al., 2019), Supervised machine learning (Souissi et al., 2017), Q-Learning (Iannucci et al., 2019b), and many more (Rose et al., 2022). Using any of these AI models usually requires many steps including data collection and preprocessing, feature extracting, model training, and feedback loop to improve the quality of the selected responses.

4.1.5. Other methods

There are alternative mathematical approaches to IRSs that are not derived from general mathematical problems. One example is REASSESS (Ossenbühl et al., 2015) that uses human-evaluated metrics and prior responses to select optimal responses. While it offers simplicity, this reliance on human evaluation can lead to inaccurate assumptions. Its mandatory learning behavior is unsuitable for automotive systems, and it lacks the option for flexible learning to enhance responses, requiring a well-established feedback loop. Another simpler approach is the cost-sensitive generic framework (Stakhanova et al., 2012; Strasburg et al., 2009), which includes steps like defining operational costs, ranking responses using a weighted sum method, and selecting the best response with an intrusion matrix. However, its reliance on static value assignments and sensitive parameters, typically defined by human experts, can make objective assessment challenging and results in potentially harmful responses.

4.2. Comparison

Table 2 summarizes all the advantages and the drawbacks of the five classes of response selection algorithms.

The primary advantage of SAW is its relative simplicity and utilization of lightweight mathematical operators, making it suitable for running on constrained devices with a polynomial run-time, without requiring complex external libraries (Bouyahia et al., 2017). However, the main drawback of SAW is the need for an adapted SAW method to achieve more accurate results. This often leads to increased complexity and longer run-time compared to the original SAW. Another drawback is the dependency on subjective parameters such as specific weights.

Table 2
Comparison of the different response selection methods.

Method	Benefits	Drawbacks
SAW	+ Simplicity and lightweight operators + Suitable for constrained devices + Polynomial run-time	- Adapted methods for accuracy increase complexity - Reliance on subjective parameters
LP	+ Flexible structures + Typically polynomial run-time + Existing libraries for solvers	- Higher complexity for modeling and calculation - Theoretically exponential run-time
Game-theoretic algorithms	+ System state consideration + Accurate system representation	- Very complex models - Computational complexity - Reliance on subjective parameters
AI-based solutions	+ Handle large amount of data + Fast response selection	- Uncertainty of the selected responses - High resource requirements
Other methods	+ Simple mathematical models + Typically fast + Combination with other methods possible + Learning is possible	- Complexity raises with large systems - Human influence has always subjective opinions

This dependency can result in highly variable outcomes that may not accurately reflect the system state (Konak et al., 2006).

A major benefit of LP is its ability to formulate a single objective function and multiple constraints, providing an accurate representation of multi-objective optimization problems. However, compared to SAW, LP requires complex implementation, resulting in increased computational complexity for large systems (Herold et al., 2017). The run-time of the algorithm depends on the solving method employed, such as the commonly used Simplex algorithm. While the Simplex algorithm has polynomial run-time for "typical" problems (Schrijver, 1998), it exhibits exponential worst-case run-time in theory (Klee and Minty, 1972).

The advantage of game-theoretic approaches lies in their consideration of the system state, resulting in a highly accurate representation of the system. Furthermore, game-theoretic approaches can be deployed in a distributed manner, as highlighted in Zonouz et al. (2014). A major drawback of this method is the use of highly complex models, which are necessary to determine optimal moves in game-theoretic algorithms. Solving such complex models often requires significant resources and leads to large communication overhead (Zonouz et al., 2014), making this approach unsuitable for constrained devices. Additionally, most models in practice make assumptions or simplifications due to the near-infinite number of possible system states (Yarygina and Otterstad, 2018; Zonouz et al., 2014; Wang et al., 2021a), as complete modeling of all states is infeasible.

Using AI-based methods is still limited because of many issues such as the high memory and computation requirements of some of these methods (Iannucci et al., 2019a) and the unrealistic responses that some models can produce (e.g., Genetic Algorithms). Additionally, uncertainty surrounding the outputs of these models limits their adoption. Finally, most of these methods rely on the availability of datasets for model training. However, autonomous vehicles often operate in dynamic and unpredictable environments. When the operating environment significantly deviates from what the AI has learned, it may encounter challenges in adapting effectively or making appropriate decisions.

Finally, while the cost-sensitive generic framework and REASSESS are simple and demonstrate promising in computer and network technologies, adapting them to a highly heterogeneous multi-bus architecture, like the vehicular reference architecture, presents significant challenges.

After careful consideration of the factors discussed above, we have chosen to explore the adapted SAW method, as well as LP with a focus on both benefit maximization and cost minimization for the design of an automotive IRS. The decision to focus on these two methods is based on their relative simplicity, computational efficiency, and their ability to accurately represent multi-objective optimization problems. The remaining algorithm families were assessed but are not pursued further due to reasons such as increased complexity, resource requirements, and limitations in modeling all possible system states.

4.3. Adopting of SAW and LP

4.3.1. Adopting of SAW

To adopt the SAW method for automotive IRSs, we first need to define the criteria $\mathcal{CR}$ that will be used to evaluate each response. For this purpose, we can utilize the HEAVENS parameters, including the cost of a response c (see Eq. (3)) and the benefit of a response b (see Eq. (4)). However, using these two parameters still presents some issues that need to be addressed in order to effectively use and adapt SAW for valid results. The first problem arises when using these parameters during the creation of the elements of the normalized decision matrix, as depicted in Eq. (5). This problem originates from the fact that our modified HEAVENS method allows values of $v_{i,j}$ to be in the set $0, 1, 10, 100$ for both criteria (i.e., c and b). If $\max_i(v_{i,j}) = 0$ applies, Eq. (5) results in an illegal operation if the criterion is a benefit. Similarly, if the criterion is a cost and $v_{a,j} = 0$, Eq. (5) also results in an illegal operation. This issue can be circumvented by using a small value greater than 0 instead of 0. The second problem does not stem from a mathematical perspective but rather from the application of this method in a fully automated IRS. Since the SAW method only considers criteria $\mathcal{CR}$ from the applicable response set $\mathcal{R}$, it does not take into account the impact I of an intrusion. As a result of this limitation, it is possible that a response incurring high costs may be chosen even for a minor intrusion. Although this is a significant challenge for the application of SAW in IRSs, this drawback has not been addressed in existing research.

To tackle this problem, it is mandatory to set the preference value p (see Eq. (6)) into relation with the intrusion impact I. For each asset A of the vehicle reference architecture and each intrusion result $\mathcal{R}$, a normalized intrusion impact can be calculated. Such a normalized intrusion impact must be calculated for each metric S, F, O, P and E of the adapted HEAVENS method in Eq. (2). This behavior is formulated in Eq. (8).

$$\alpha_{\{S,F,O,P,E\},A,\mathcal{R}} = \begin{cases} \frac{w_{\{S,F,O,P,E\},A,\mathcal{R}} \cdot v_{\{S,F,O,P,E\},A,\mathcal{R}}}{\sum_{|\mathcal{R}|}(w_{\{S,F,O,P,E\},A} \cdot v_{\{S,F,O,P,E\},A})}, & \text{if } \sum_{|\mathcal{R}|}(w_{\{S,F,O,P,E\},A} \cdot v_{\{S,F,O,P,E\},A}) \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

Similar to Eq. (6), a weighted sum must be calculated. But, since the individual weights w are already included in Eq. (8), a simple summation over all metrics S, F, O, P and E of the adapted HEAVENS method is sufficient. This sum will be set into relation with the preference value of the responses from Eq. (6), such that the response r_i with the highest preference value p will be used, which is below the sum of all normalized HEAVENS values as depicted in Eq. (9).

$$\text{best response} = \max \left\{ p_i \mid p_i < \rho \cdot \sum_{l \in \{S,F,O,P,E\}} \alpha_{l,A,\mathcal{R}} \right\} \tag{9}$$

The parameter ρ in Eq. (9) is a parameter to adjust larger deviations in the order of magnitude between the sum of the normalized HEAVENS and the preference value p.

4.3.2. Adopting of linear programming

The first step to adopt the LP is defining the objective function. For the set of possible responses $\mathcal{R}$, it is possible to define two different objective functions:

- The first option of an objective function follows the principle of maximum benefit as depicted in Eq. (10). The goal is to solve the binary decision vector $\vec{s}$ to maximize the benefit b. Although this can lead to very good solutions, it is possible that the best executable response is not found immediately since preconditions of identified responses are not satisfied.

$$\sum_{i=1}^{|\mathcal{R}|} s_i b_i \rightarrow \max \tag{10}$$

- The second option of an objective function follows the minimum cost principle and is comparable to existing IRSs (Herold et al., 2017; Herold, 2017). Eq. (11) therefore leads to more conservative responses since the cost c will be minimized and the benefit b of a response is not considered. A drawback is that the identified solution inside $\vec{s}$ might not heal the system completely and another try might be necessary.

$$\sum_{i=1}^{|\mathcal{R}|} s_i c_i \rightarrow \min \tag{11}$$

For both objective functions from Eqs. (10) and (11) the same constraints must be satisfied for a response to qualify for execution. Existing constraints of IRSs using LP (Herold et al., 2017; Herold, 2017) are not suitable for an automotive IRS. Because of that, specific constraints must be elaborated:

1. The cost c of the response must be below the impact I of the detected intrusion (Herold et al., 2017). Eq. (12) depicts this first constraint.

$$\sum_{i=1}^{|\mathcal{R}|} s_i c_i < I \tag{12}$$

2. Only one response can and must be executed as depicted in Eq. (13).

$$\sum_{i=1}^{|\mathcal{R}|} s_i = 1 \tag{13}$$

It is additionally necessary that $\vec{s}$ is a binary vector, leading to the variable definition $s_i \in \{0, 1\}$.

5. Proposed automotive IRS

In this section, we will discuss some design decisions regarding REACT, our proposed automotive IRS (refer to Section 5.1) and detail its components (refer to Section 5.2).

5.1. IRS deployment

Our proposed automotive IRS can be deployed in three different locations:

- Central Gateway: The vehicle will have one IRS that receives information from various ECUs. This central IRS will have a comprehensive view and understanding of the entire system. However, it is considered a single point of failure.
- Domain Gateway: The vehicle will have one IRS per domain gateway. Each one will be mainly responsible for the ECUs belonging to that domain and will interact with other IRSs. Implementing this solution requires the existence of an Intrusion Response eXchange Protocol (IRXP) (Hamad et al., 2021).
- ECU: The vehicle will have one IRS per ECU. This IRS will be primarily responsible for reacting to attacks related to its host ECU. Simultaneously, it can exchange responses related to other ECUs if needed. Choosing this option ensures the absence of a single point of failure. However, deploying such a solution requires that each ECU is capable of running the IRS, and it also necessitates the existence and the support of an IRXP (Hamad et al., 2021).

The architecture depicted in Fig. 4 illustrates the scenario where the IRS is deployed in the central gateway. Any potential change would be primarily associated with the source of certain information required for the functionality of the IRS, whether it originates from the same ECU (in the case of implementing the IRS per ECU) or from external sources such as other ECUs or domains at the gateway. Regardless of the chosen deployment location for the IRS, it necessitates the reception and sharing of information with other components within the vehicle, as outlined below:

- Attack Information: This information is provided by the IDS, and as described in 3.1.1, it includes the source of the attack, the destination, the intrusion result, and the impact of the attack. Recent IDSs, such as Jeong et al. (2024) and Ding et al. (2024), are capable of identifying the source and destination of an intrusion using various technologies, such as CAN databases (used by Jeong et al. (2024)) or ECU fingerprinting (Cho and Shin, 2016; Kneib and Huth, 2018). The intrusion impact can be calculated as described in 3.1.2. Additionally, the intrusion result can be derived from the attack type, which existing IDSs, such as Han et al. (2021), can provide. In our research, we consider the IDS functionality as trusted, treating it as a blackbox that reliably detects intrusions without requiring additional false-positive handling (Herold et al., 2016; Ullah et al., 2022). In our architecture, we place the IDS in the domain gateway. Consequently, a security sensor (Anwar et al., 2017) is needed to monitor its portion of the environment for security-related observations. This data is then reported to the domain-specific gateway, which houses the domain IDS.
- Status Information: This includes information about the various states of the vehicle and its surroundings. This data is collected and aggregated from various vehicle sensors and shared with the IRS.
- Response Information: This information can encompass the precise responses needed for specific ECUs or those that need to be shared with the SOC. In our architecture, we assume the presence of response agents located in each ECU. These agents are responsible for receiving responses and deploying them within the respective ECU.

It is crucial to mention the necessity of ensuring the security of this data by implementing secure communication between the ECU, domain gateway, and the IRS.

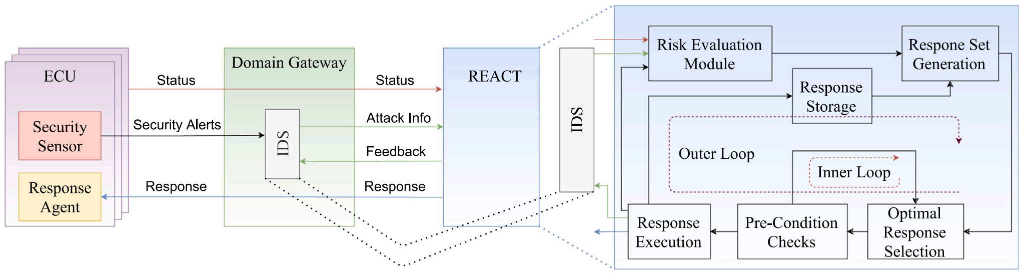


Fig. 4. Internal architecture of REACT.

5.2. IRS component

The IRS consists of the following sub-components (as shown in Fig. 4):

- Risk Evaluation Module: This module will be responsible for assessing the impact of an intrusion. The component will receive information about the intrusion from the IDS as well as information about the vehicle status.
- Response Set Generation: This module compiles a list of possible responses, utilizing information obtained from both the IDS and the risk evaluation module. Please note that not every response is applicable to every type of intrusion result (refer to Table 1).
- Optimal Response Selection: This component integrates data from all previous modules to determine the optimal response that can be applied. Within this component, any of the algorithms presented in Section 4.1 can be integrated.
- Precondition Checking: Given the limitations imposed by the system architecture, where not all types of responses can be applied (for example, in cases where a sensor is unavailable due to a DoS attack, it may not always be possible to use a redundant source of information from another sensor if such a backup sensor does not exist), it is imperative to verify whether the selected optimal response is applicable or if an alternative response must be chosen. The Precondition Checking module receives the chosen response and assesses its feasibility. If a response is found to be inapplicable, a feedback loop is established with the previous Optimal Selection Module. This *inner loop* is repeated until the necessary preconditions for an individual response are met. The order of the Optimal Response Selection and the Precondition Checking is carefully evaluated and results in time benefits:

  1. "Check-First-Then-Select": The logical order of first eliminating all inapplicable responses and subsequently selecting the best response r from the remaining available options is illustrated by the timing behavior of Eq. (14).

  $$t = \left(\sum_{i=1}^{|\mathcal{R}|} t_{check,r_i}\right) + t_{select,r} + t_{execute,r} \tag{14}$$

  The time to select the optimal response $t_{select,r}$ and the time to execute the response $t_{execute,r}$ are summed only once, since the selected response will satisfy the preconditions. In contrast, the time to check the preconditions $t_{check,r}$ is summed over the set of possible responses $\mathcal{R}$, since every response's precondition will be checked.
  2. "Select-First-Then-Check": While a response may be applied with the probability p, it might also be that the constraints are not satisfied with a probability $(1-p)$. This leads to a timing behavior of Eq. (15).

  $$t = t_{select,r_1} + t_{check,r_1} + p \cdot t_{execute,r_1} + (1-p) \cdot \sum_{i=2}^{|\mathcal{R}|} \left(t_{select,r_i} + t_{check,r_i}\right) \tag{15}$$

  While the first selected response must always be checked, it is only executed with the probability p. If the preconditions are not satisfied, the *Inner Loop* will be repeated maximum $|\mathcal{R}| - 1$ times.

  It is evident that for a certain number of responses approaching infinity, Eqs. (14) and (15) yield the same runtime t when $p = 0.5$. For higher values of p, the runtime as per Eq. (15) is even lower. This holds true even when $t_{select,r}$ decreases, as the number of possible responses decreases accordingly. Based on these equations, the architecture depicted in Fig. 4 exhibits a "Select-First-Then-Check" behavior.
- Response Execution: This component is responsible for transmitting the chosen response initially to the domain-specific gateways and subsequently to the respective ECUs for implementation through their local response engines. After a predefined duration, this component triggers the IDS to assess the effectiveness of the applied response in mitigating the intrusion. By incorporating this IDS-Feedback loop, the *Outer Loop* can be iterated multiple times, each iteration involving a system re-evaluation. This concept serves to counter persistent attacks or stepping-stone attacks effectively. Furthermore, the feedback loop can be utilized to update the parameters of the risk evaluation module for addressing future intrusions.
  An essential consideration in the IRS architecture shown in Fig. 4 is the implementation of termination criteria for the *inner* and *outer loop*. The absence of such criteria could lead to an endless loop, posing a risk to the stability of the entire IRS system. While some prior research has addressed termination criteria (Hamad et al., 2021; Shameli-Sendi et al., 2012), these methods often involve complex evaluation techniques (Cardellini et al., 2022; Iannucci et al., 2021) or rely on artificial intelligence support (Lopes and Hutchison, 2020). However, the high computational requirements and intricate modeling approaches associated with these methods are impractical for automotive infrastructure. To address the challenge of preventing endless loops in both the *inner* and *outer* loops, we employ two distinct methods.

  1. Preventing Inner Endless Loops: To avoid an endless evaluation of preconditions, we continuously reduce the possible response set by eliminating non-applicable responses. Additionally, we have introduced a special response, labeled as "No Action" (indexed as 31), which will consistently lead to the last possible response. This specific response carries the highest cost, similar to the impact of an intrusion, but provides no benefit. These attributes ensure that the *inner loop* never reaches a deadlock since "No Action" can always be applied.
  2. Avoiding Outer Endless Loops: Once a response is applied, the system undergoes an analysis through the IDS-Feedback mechanism to identify if a new stepping-stone attack is detected or if the system is secure. In case a

new stepping-stone attack is detected, the entire *outer loop* illustrated in Fig. 4 reiterates. To prevent an endless loop scenario when the same response is repeatedly applied, we implement changes to the parameters of the applied response based on the success of the response. The parameter adaptation differs between a successful and a non-successful response. When the selected response is unsuccessful, it indicates that the benefit values assigned to all HEAVENS parameters may not be accurate. Consequently, an adjustment is needed, resulting in a reduction of the benefit values for all HEAVENS parameters in the previously applied response. This entails the assumption that the relative order of each parameter remains unchanged; for example, if the safety benefit held a higher value than the financial benefit prior to the adjustment, it will continue to do so afterward. This behavior is mathematically expressed in Eq. (16).

$$\forall i \in \{S, F, O, P\}:$$

$$i_{\text{new}}(i_{\text{old}}) = \begin{cases} 10, & \text{if } i_{\text{old}} = 100 \\ 1, & \text{if } i_{\text{old}} = 10 \\ 0, & \text{if } i_{\text{old}} = 1 \text{ or } i_{\text{old}} = 0 \end{cases} \tag{16}$$

A similar parameter adaptation is required in case the response was applied successfully. However, the parameters cannot simply be increased, as this could lead to predictable responses. Predictable responses pose security risks, as attackers can exploit this behavior (Bouyahia et al., 2017). For that reason, two adaptations are made if the response is successful to avoid predictable behavior:

- Original values are restored if the response was previously not successful and its values were adapted according to Eq. (16).
- In a second step, the corresponding weights $w_{i \in S,F,O,P}$ are randomly adjusted using a prefactor r, where $r_{min} \le r \le r_{max}$. This retains the original order of magnitude of w_i while introducing sufficient variation through the multiplication $r \cdot w_i$ to generate different results in the next iteration.

As previously mentioned, the parameters to calculate the intrusion impact (Eq. (2)), the response cost (Eq. (3)) and the response benefit (Eq. (4)) rely on input by security experts. However, this input may not always be optimal (Lautenbach et al., 2021). Consequently, this can lead to the selection of an undesired response. Fortunately, the *outer loop* provides a mechanism to compensate for potentially incorrect parameters. In cases where responses prove ineffective, the parameters are dynamically adapted using Eq. (16).

Note that Eq. (16) presented earlier does not account for the dynamic environmental parameter, denoted as E, and its corresponding weight, w_E. Further details and definitions are necessary to incorporate this parameter into the adaptation process. These details should encompass various aspects of the vehicle's status and its surrounding environment. For simplicity, we have focused on the vehicle's velocity as a parameter that can help represent the vehicle's status. To determine a realistic rating for the impact of vehicle speed, several factors must be taken into account. Studies of traffic accidents have revealed that the impact is influenced not only by the types of vehicles involved but also by their positions at the potential crash site (Jurewicz et al., 2016). Additionally, the age of the passengers in the vehicles can affect the impact of injuries in a traffic accident (Richards, 2010). Based on this research, the approach presented in Eq. (17) is applied to the parameter E in the adapted HEAVENS method's prototype implementation (Jurewicz et al., 2016; Richards, 2010).

$$E(v) = \begin{cases} 100, & \text{if } v \ge 75 \text{ km/h} \\ 10, & \text{if } 50 \text{ km/h} \le v < 75 \text{ km/h} \\ 1, & \text{if } 30 \text{ km/h} \le v < 50 \text{ km/h} \\ 0, & \text{if } 0 \text{ km/h} \le v < 30 \text{ km/h} \end{cases} \tag{17}$$

- Response Storage: Within this component, a repository is maintained containing a range of potential responses alongside their associated metrics. These metrics can be updated through the feedback mechanism or expanded with the inclusion of new responses and parameters via an external connectivity interface. When implementing this on specific hardware, it is crucial to implement security measures to prevent unauthorized tampering with the memory area.

Table 3
IDS-related information and vehicle state parameters for both evaluation scenarios.

Property	Scenario 1	Scenario 2
Name	Adversarial sample	Information disclosure at the infotainment system
Infected asset	Front camera	Infotainment gateway
Affected asset	Acceleration control	Infotainment gateway
Intrusion result	Falsify/Alter behavior	Information Disclosure
Dynamic parameter	Velocity: 70 km/h	Velocity: 0 km/h

Our proposed IRS architecture, featuring both an *inner loop* and an *outer loop*, coupled with the incorporation of automotive-specific considerations into the external architecture, introduces a novel paradigm in the realm of fully automated IRSs. Note that there is already some related work for each part of the IRS (such as the selection method), which was covered in the previous sections. However, there is no system that attempts to include all the aspects against which we can compare our work.

6. Evaluation

6.1. Implementation, testbed, and use cases

The proposed IRS was implemented using the Python programming language. To implement Linear Programming and the associated Simplex algorithm, we utilized the `PuLP library` (Mitchell et al., 2011), a well-established choice, along with the GNU Linear Programming Kit as the solver. It is important to note that the adapted SAW method remains independent of this decision, as it relies solely on standard Python mathematical operators.

The testbed designed for evaluating the IRS incorporates an embedded system setup to realistically emulate the automotive infrastructure. To ensure this fidelity, our implementation was executed on a Raspberry Pi 4 Model B Rev 1.2, a choice justified by the device's ARM-based quad-core processor running at 1.5 GHz. This processing power closely aligns with the high-performance chips commonly found in the automotive industry.

The goal of the evaluation is to assess two key aspects of the proposed IRS. Firstly, we aim to evaluate its proficiency in optimal response selection, and secondly, we intend to measure various computational metrics, including memory consumption and the time required to obtain optimal responses while using the three different selection algorithms: LP with maximum benefit, LP with minimum cost, and adapted SAW.

For our evaluation, we employed two representative intrusion scenarios inspired by real-world intrusions:

1. Adversarial Sample: This scenario involves slight modifications to the input data of a machine learning algorithm, resulting in significantly different outputs from the original (Mahima et al., 2021). Given the prevalent use of machine learning algorithms in cameras for automated vehicles, they are vulnerable to exploitation via adversarial samples (Mahima et al., 2021). In our evaluation, we exploited a front camera in a rural setting, leading to an altered behavior in the acceleration control.
2. Information Disclosure at the Infotainment System: This scenario draws inspiration from an actual attack on a vehicle, where an information disclosure in the infotainment system served as the initial step in a stepping-stone attack (Miller and Valasek, 2015).

The specific IDS parameters and vehicle states employed as input for the scenarios are meticulously detailed in Table 3. Please remember that in our prototype of the IRS, we consider only the velocity of the attacked vehicle as an illustrative example of a vehicle's status.

6.2. Results

In this section, we will present the results of testing our IRS using two prominent scenarios. We will evaluate response quality, response selection time, memory consumption, and the adaptation of response parameters for each of the three selection algorithms: LP with maximum benefit, LP with minimum cost, and the adapted SAW.

6.2.1. Response quality

The objective of the response quality evaluation is to assess how different optimal selection algorithms prioritize responses and determine the overall impact and benefit of the applied responses. To achieve that, the precondition of each response is set to 'rejected' for every proposed response. This ensures that the IRS will continue to suggest responses from the list of possible responses. Each applied response can have both positive and negative effects on the system, so the cost and benefit values of the selected responses are presented. In this evaluation, default parameters are utilized for each new test, ensuring uniformity in the algorithm evaluation across various metrics.

Fig. 5 depicts the cost and benefit of all proposed responses in the order they are applied by the respective algorithm for both scenarios. The figure shows that our proposed IRS suggests a different number and order of responses for various scenarios and for different selection algorithms within the same scenario. Please note that the figure shows that some responses were selected twice. For example, the response of restarting the misbehaving system (indexed with number 19, see Table 1), was selected twice. However, it is important to clarify that the response was selected for different systems. In other words, the first restart is related to the camera, while the second is for the acceleration control. In addition, as expected and shown in Figs. 5(a) and 5(b), the LP method with maximum benefit starts at very high benefits. Similarly, the LP with minimum response costs starts at a very low cost and more expensive responses are not selected until later stages, as shown in Figs. 5(c) and 5(d). Notably, the LP with maximum benefit operates independently of the cost. However, it always ensures that the cost of the response is less than the impact of the intrusion (see Eq. (12)).

The reason for the arbitrary behavior is that Linear Programming only follows one optimization function and just satisfies the constraints, but does not sort by constraints. Similarly, LP with minimum cost delivers arbitrary values with respect to the benefit because it only considers cost metrics in its optimization. While the LP with the minimum cost provides more conservative solutions, the LP with maximum benefit suggests more offensive solutions. In a real-world scenario, LP with minimum cost might require multiple responses since its benefits are arbitrarily sorted, while LP with maximum benefit might require more iterations of the "inner loop" since the preconditions for more offensive responses might not be fulfilled.

The adapted SAW method exhibits a similar arbitrary behavior as shown in Figs. 5(e) and 5(f). However, it is noticeable that adapted SAW may select responses with a cost higher than the impact of the intrusion (see Fig. 5(f)). Given that the adapted SAW method does not consider constraints, it is an unattractive solution to use any SAW method in an automatic IRS.

Table 4
Memory consumption of the IRS in kB using static evaluation.

	LP with max benefit	LP with min cost	Adapted SAW
Scenario 1	19 308	19 206	11 296
Scenario 2	19 228	19 344	11 220

6.2.2. Time of response selection

To evaluate the time required for selecting a response from a given response list using the selection algorithms, we utilized the previously described method where the *inner loop* of the IRS repeats multiple times. It is important to note that the generation of the response set occurs only once for an individual intrusion. The time required for list generation is independent of the selection algorithm, measuring at 4.32 ms for scenario 1 and 3.82 ms for scenario 2. The difference in the measured time between the scenarios is due to the variation in number of possible responses.

Fig. 6 illustrates the time consumed by the three selection algorithms during the process of selecting different responses. Please note that the X-axis represents the order of the response, not the index of the response. The figure indicates that the adapted SAW method consumes less time compared to the LP methods. Specifically, the LP method with maximum benefit typically consumes more time due to the need for multiple iterations, as its offensive responses may not meet necessary preconditions. Slightly less time is needed for the LP method with minimum cost, although its conservative responses are selected after fewer precondition checks. Overall, all algorithms demonstrate good performance on a resource-constrained embedded system.

6.2.3. Memory consumption

To measure memory consumption, we utilized Python's internal `resource` module (Python Software Foundation, 2022). Since some of the optimal selection algorithms rely on third-party libraries, the assessment of memory consumption includes the memory allocated for these functionalities as well. The results are presented in Table 4. The results show that both LP with maximum benefit and LP with minimum cost methods consume nearly the same amount of memory, while the adapted SAW method exhibits considerably lower memory consumption. This difference can be attributed to the external libraries `PuLP` and the `GNU Linear Programming Kit`, which require more memory due to their complex data structures and solving methods. Nevertheless, all three selection algorithms exhibit low memory consumption, making them suitable for use in resource-constrained embedded hardware systems.

6.2.4. Dynamic evaluation

The dynamic evaluation concentrates on two key aspects: response and threat impact parameters adaptation (refer to Section 3) and the inclusion of velocity considerations (as shown in Eq. (17)). When it comes to parameters adaptation, response quality is assessed based on their cost and benefit. In terms of velocity, we evaluate response variation. These assessments are conducted for both scenarios 1 and 2. By testing all three implemented optimal selection algorithms, we can compare their dynamic behavior.

Parameters adaption. To assess the impact of changing parameters, we conducted two repetitions of each scenario, each comprising five iterations of the *outer loop*. In one set of iterations for each scenario, we consistently deemed the responses as successful, while in the other set of five iterations, the responses were uniformly considered unsuccessful. The benefits and costs of the five optimally selected responses for

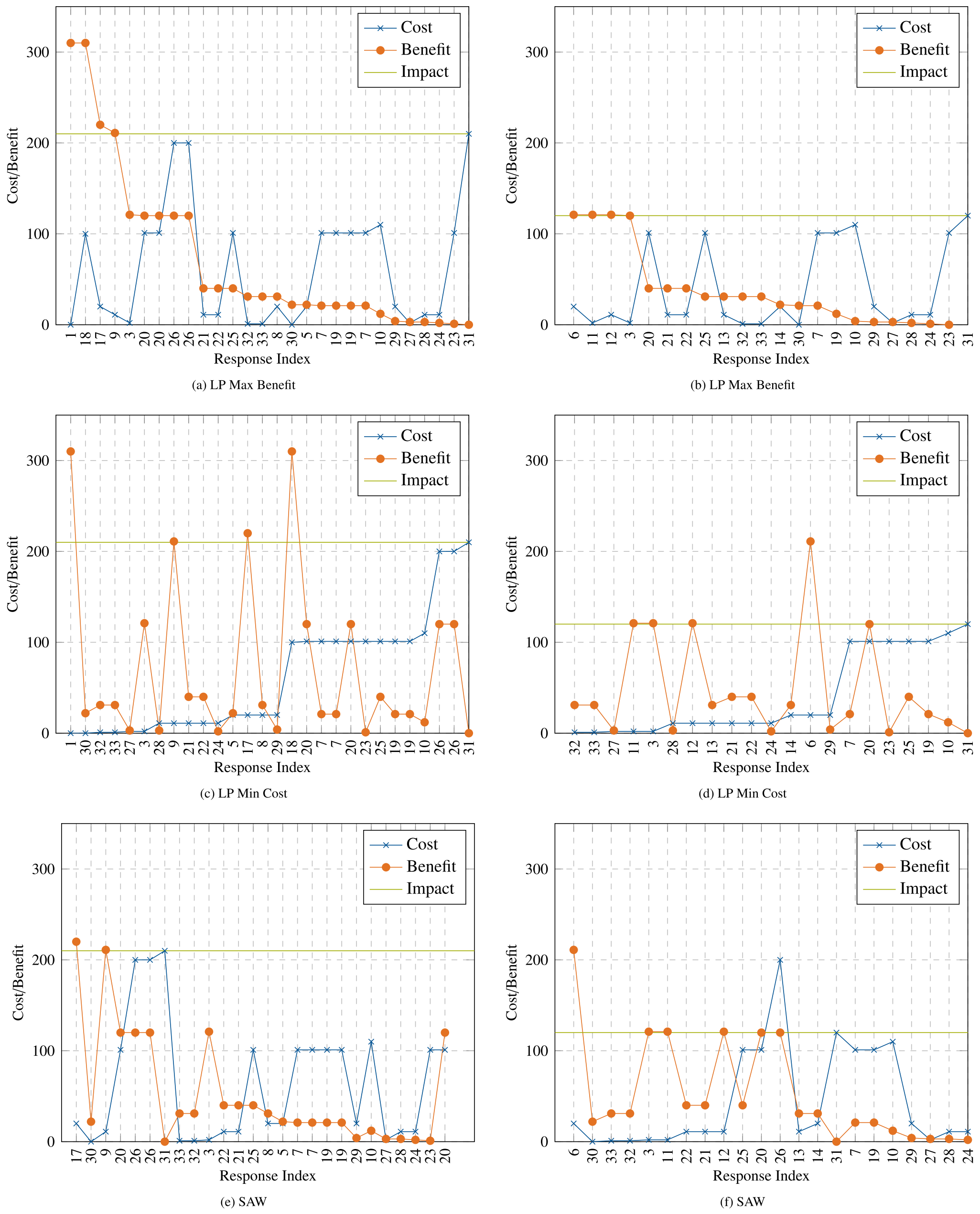


Fig. 5. Evaluation of the response benefit and cost for Scenario 1 (left) and Scenario 2 (right) using LP with maximum benefit (top), LP with minimum cost (middle), and adapted SAW (bottom).

both scenarios, as determined by the three selection algorithms, under the assumption that the responses were always successful, are presented in Fig. 7. Correspondingly, the results under the assumption that the responses were consistently unsuccessful are displayed in Fig. 8.

In consistently successful attacks, we observed that parameter weights change within the range of ±20% (we have selected $r_{min} = 0.8$ and $r_{max} = 1.2$). The purpose of these changes was to reduce response predictability. In both scenarios, changes in response benefit were evident. However, in the first scenario, all three algorithms retained the same response as shown in Figs. 7(a), 7(b), and 7(c). This was changed in the second scenario, where responses were altered for the LP with maximum benefit and adaptive SAW algorithms as shown in Figs. 7(d) and 7(f). The reason for the absence of changes in the selected responses in the first scenario when using LP with maximum benefits or adapted SAW algorithms can be attributed to the specific response chosen: transitioning to a safe mode (indexed with 17). This

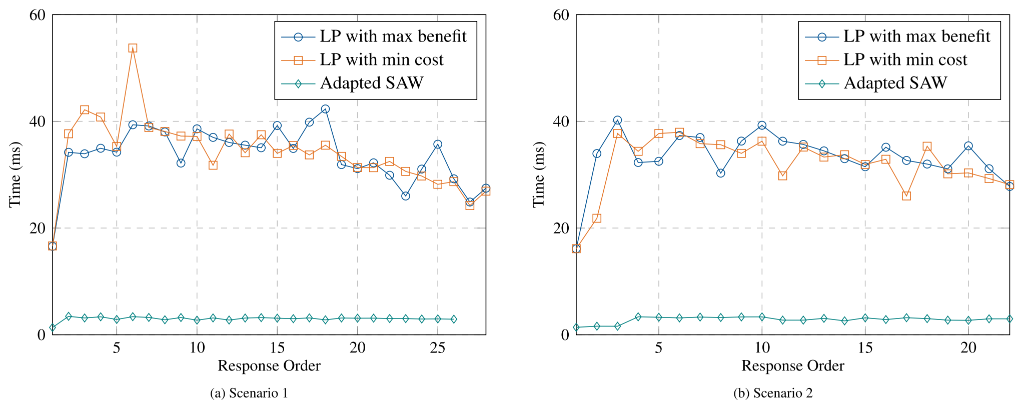


Fig. 6. Evaluation of consumed time for response selection using the three selection algorithms for both scenarios.

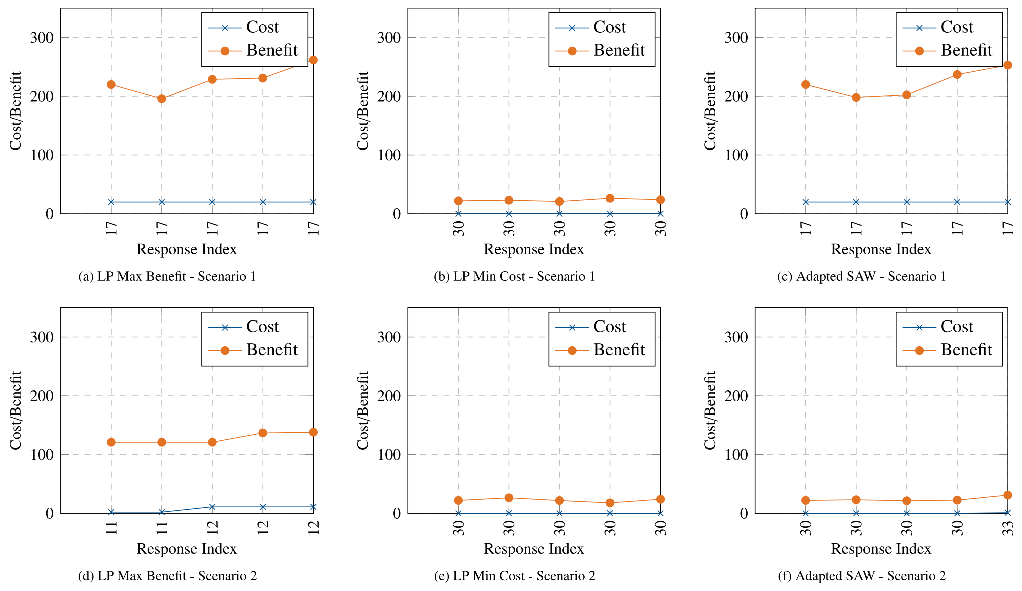


Fig. 7. Evaluation of parameter adaptation in Scenario 1 (top) and Scenario 2 (bottom) for the responses selected over five iterations using the three selection algorithms, assuming the responses were consistently considered successful.

response had very high benefit values, as determined through the initial evaluation process, making minor variations of ±20% inconsequential to the overall result. Consequently, minor variations of ±20% did not affect the overall result, as the next possible response had significantly lower benefit values. To avoid such a constant behavior, a more substantial modification of the response parameters or the use of an asymmetric window for the prefactor, with a higher probability of negative values, can be implemented. Notably, the LP method with minimum cost (Figs. 7(b) and 7(e)) did not consider response benefits in its optimization function, rendering modifications to response benefit irrelevant. This method-related limitation persisted across both simulated scenarios.

In the case of consistently unsuccessful attacks, we observe more substantial variations in the selected responses compared to the previous case (see Fig. 8). This behavior is expected, as the parameter adaptation in a non-successful case involves higher orders of magnitude, as shown in Eq. (16), compared to the successful case. Similar to the previous analysis, the LP method with minimum cost optimization consistently generates the same response due to the exclusion of response benefit in the optimization process, as shown in Figs. 8(b) and 8(e). Conversely, LP with maximum benefit optimization aligns with expectations. Although the initial response is similar to the successful case, subsequent responses exhibit lower benefits (Figs. 8(a) and 8(d)) and higher costs as a side effect. Notably, response index 26 (killing the process) appeared twice in Figs. 8(a) and 8(c), each referring to different components (i.e., camera and acceleration control). The adapted SAW method consistently produces varying results with less distinct trends in benefit and cost when compared to LP with maximum benefit (Figs. 8(c) and 8(f)). This observed behavior holds true for both scenarios1 and 2, underscoring the expected functionality of parameter adaptation for non-successful cases.

In conclusion, this assessment of dynamic parameter adaptation confirms that LP with maximum benefit and the adapted SAW methods perform effectively with adjusted parameters, rendering the results

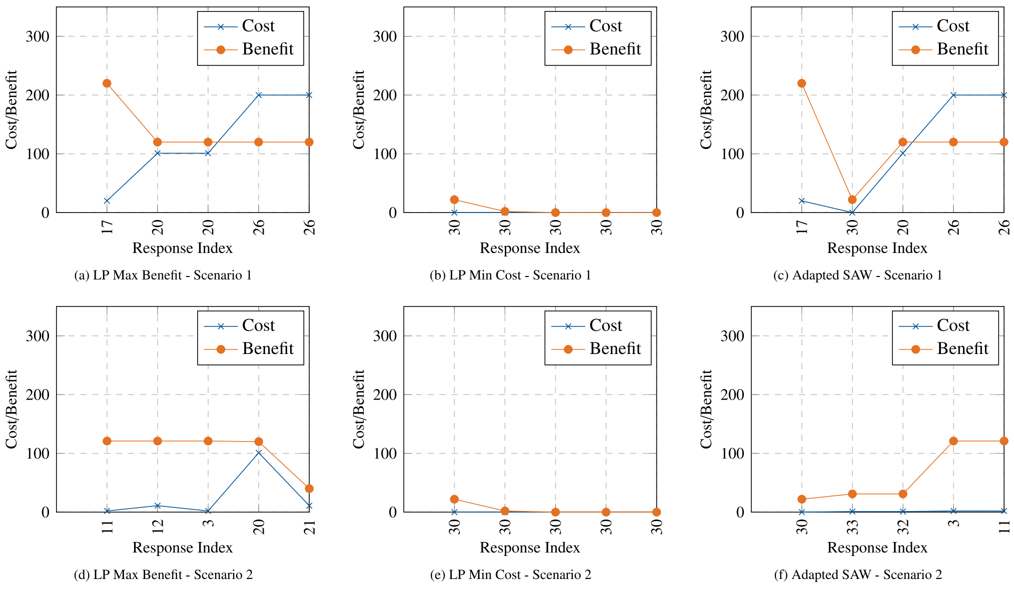


Fig. 8. Evaluation of parameter adaptation in Scenario 1 (top) and Scenario 2 (bottom) for the responses selected over five iterations using the three selection algorithms, assuming the responses were consistently considered unsuccessful.

Table 5
Impact of the velocity for the evaluated scenarios, using Eq. (2).

	Impact (unitless)		
	0 km/h	50 km/h	100 km/h
Scenario 1	200	210	300
Scenario 2	120	130	220

valid for both test cases. On the other hand, the LP method with minimum cost optimization falls short in its capacity to respond to parameter shifts in response benefit values. Consequently, this method appears less appealing for identifying optimal responses in autonomous IRS.

Inclusion of velocity considerations. The second key aspect of dynamic evaluation involves assessing the influence of vehicle velocity on the selected responses. In our current prototype system, the environmental parameter E is treated similarly to other HEAVENS parameters in Eq. (2), as their respective weights w are either one or zero. As we alter the velocity, the environmental parameter for an intrusion takes on different values, as indicated in Eq. (17). Therefore, intrusion's impact is more significant at higher velocities. For this test, both scenario one and two are assessed at three velocities: 0, 50, and 100 km/h, using all three implemented algorithms, with each evaluation beginning with the default data-set.

While the intrusion impact calculation in Table 5 functions as expected, each algorithm consistently selects the same response within each scenario, regardless of the velocity. This behavior can be attributed to the high impact values in the two evaluated scenarios. In cases of less severe intrusions or during the early stages of a stepping-stone attack, where the HEAVENS parameters result in lower values, the velocity's impact becomes relatively more substantial, thus leading to varying results. Nonetheless, it is important to emphasize that the proposed IRS architecture is adaptable since the individual weights w for HEAVENS parameters can be customized as per Eq. (2). This customization minimizes the over-representation of static HEAVENS parameters, enabling the velocity to exert a more pronounced influence on the selected response.

6.2.5. *Final remarks*

The evaluation of the developed IRS reveals the advantages and drawbacks of each selection method. The adapted SAW method is limited by its inability to consider constraints. Consequently, it is not feasible to employ this method in a fully automated IRS. On the other hand, LP with minimum cost consistently favors constant responses and is, therefore, unsuitable for optimal response identification. Despite its successful application in existing research (Herold et al., 2017; Herold, 2017), the results demonstrate suboptimal behavior for the automotive use case. Nevertheless, it is well-suited for proposing follow-up responses once the primary intrusion has been mitigated. These follow-up responses can enhance security by alerting a SOC and providing information to the car manufacturer, ultimately leading to updated software. In contrast, the LP method with maximum benefit, excels in all metrics evaluated for an automotive IRS. Since it offers responses with high benefits from the outset, it is well-suited to respond to the primary intrusion.

7. Conclusion and outlook

Modern vehicles' intricate architecture and advanced connectivity present unique intrusion challenges. While automotive security research has traditionally emphasized IDSs as a secondary defense layer, the development of vehicle IRS is in its early stages, drawing inspiration from related industries. To delve into the development of an automotive IRS, we sought answers to three key questions: defining potential responses, outlining response evaluation criteria, and optimizing response selection. Initially, we categorized automotive intrusions and stepping-stone attacks into five distinct categories to create a more versatile intrusion model. Similarly, we classified responses, creating a formal description for both intrusions and responses. Additionally, we investigated necessary adjustments to existing risk assessment models to support response evaluation. Furthermore, we conducted a comprehensive comparison of various optimal selection algorithms, highlighting the adaptability of the SAW method and Linear Programming (LP) with various optimizations for IRS integration. Although other algorithm families may gain relevance in the future, they currently face limitations in the automotive context. In addition to these findings, we proposed an IRS architecture that accommodates the distributed nature of vehicles and addresses automotive-specific constraints. Evaluation in real-world scenarios has led to the development of a novel vehicular IRS, demonstrating its potential for integration into modern distributed vehicle architectures and enhancing overall security.

While the focus of the paper is on the analysis and design of the IRS, the implementation of the external architecture and the response execution modules on the local engines on each ECU is still a challenge towards an IRS as a system. To test such an overall IRS system, real-world data sets, including both normal operation and attack scenarios, are needed. Extensive evaluation in Software-in-the-Loop or Hardware-in-the-Loop testbeds can extend the existing evaluations of algorithms and the overall system. With respect to the secure communication of intrusions and responses, further research and standardization are needed to be performed in order to ensure that the developed IRS does not only reply in an adequate manner but also distributes its responses. In this direction, leveraging existing efforts such as International Telecommunication Union (2022) and Matthews and Feinstein (2007) by extending them towards establishing a standardized method for securely exchanging the proposed responses within the vehicle and with other vehicles would provide a solid foundation, as these existing standards and guidelines already offer valuable insights. Also, it is important to note that the functionality of our proposed system depends on the availability of information about the attack, such as its source, destination, and type, which needs to be provided by the IDS. This information can be obtained by integrating existing research approaches, as demonstrated in Jeong et al. (2024) and Ding et al. (2024). Finally, the modular architecture of REACT allows an easy extension towards more complex vehicle architectures and new intrusions or responses. Additionally it allows the integration of new selection algorithms in the future to adapt to possible changed needs.

CRediT authorship contribution statement

Mohammad Hamad: Writing – review & editing, Writing – original draft, Supervision, Project administration, Methodology, Funding acquisition, Conceptualization. **Andreas Finkenzeller:** Writing – review & editing, Methodology. **Michael Kühr:** Software, Methodology, Investigation, Conceptualization. **Andrew Roberts:** Writing – review & editing, Writing – original draft, Validation. **Olaf Maennel:** Writing – review & editing, Writing – original draft, Validation. **Vassilis Prevelakis:** Writing – review & editing, Validation, Methodology. **Sebastian Steinhorst:** Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Mohammad Hamad reports financial support was provided by European Union. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work is supported by the European Union-funded projects CyberSecDome (Agreement No.: 101120779).

References

Alrefaei, F., Alzahrani, A., Song, H., Alrefaei, S., 2022. A survey on the jamming and spoofing attacks on the unmanned aerial vehicle networks. In: 2022 IEEE International IOT, Electronics and Mechatronics Conference. IEMTRONICS, IEEE, pp. 1–7.

Anuar, N.B., Papadaki, M., Furnell, S., Clarke, N., 2012. A response strategy model for intrusion response systems. In: Gritzalis, D., Furnell, S., Theoharidou, M. (Eds.), Information Security and Privacy Research. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 573–578.

Anwar, S., Mohamad Zain, J., Zolkipli, M.F., Inayat, Z., Khan, S., Anthony, B., Chang, V., 2017. From intrusion detection to an intrusion response system: Fundamentals, requirements, and future directions. Algorithms 10 (2), http://dx.doi.org/10.3390/a10020039.

Anwar, S., Zain, J.M., Zolkipli, M.F., Inayat, Z., Jabir, A.N., Odili, J.B., 2015. Response option for attacks detected by intrusion detection system. In: 2015 4th International Conference on Software Engineering and Computer Systems. ICSECS, pp. 195–200. http://dx.doi.org/10.1109/ICSECS.2015.7333109.

AUTOSAR, 2020. Specification of Intrusion Detection System Protocol. Technical Report, AUTOSAR Consortium, URL: https://www.autosar.org/fileadmin/standards/R20-11/FO/AUTOSAR_PRS_IntrusionDetectionSystem.pdf.

Barletta, V.S., Caivano, D., Vincentiis, M.D., Ragone, A., Scalera, M., Martín, M.Á.S., 2023. V-soc4as: A vehicle-soc for improving automotive security. Algorithms 16 (2), 112.

Bashendy, M., Tantawy, A., Erradi, A., 2023a. Intrusion response systems for cyber-physical systems: A comprehensive survey. Comput. Secur. 124 (C), http://dx.doi.org/10.1016/j.cose.2022.102984.

Bashendy, M., Tantawy, A., Erradi, A., 2023b. Intrusion response systems for cyber-physical systems: A comprehensive survey. Comput. Secur. 124, 102984.

Bouyahia, T., Cuppens-Boulahia, N., Cuppens, F., Autrel, F., 2017. Multi-criteria recommender approach for supporting intrusion response system. In: Cuppens, F., Wang, L., Cuppens-Boulahia, N., Tawbi, N., Garcia-Alfaro, J. (Eds.), Foundations and Practice of Security. Springer International Publishing, Cham, pp. 51–67.

Cardellini, V., Casalicchio, E., Iannucci, S., Lucantonio, M., Mittal, S., Panigrahi, D., Silvi, A., 2022. An intrusion response system utilizing deep Q-networks and system partitions. http://dx.doi.org/10.48550/ARXIV.2202.08182, https://arxiv.org/abs/2202.08182.

Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S., Koscher, K., Czeskis, A., Roesner, F., Kohno, T., 2011. Comprehensive experimental analyses of automotive attack surfaces. In: 20th USENIX Security Symposium (USENIX Security 11).

Chevalier, R., Plaquin, D., Dalton, C., Hiet, G., 2019. Survivor: A fine-grained intrusion response and recovery approach for commodity operating systems. In: Proceedings of the 35th Annual Computer Security Applications Conference. ACSAC '19, Association for Computing Machinery, New York, NY, USA, pp. 762–775. http://dx.doi.org/10.1145/3359789.3359792.

Cho, K.-T., Shin, K.G., 2016. Fingerprinting electronic control units for vehicle intrusion detection. In: 25th USENIX Security Symposium (USENIX Security 16). pp. 911–927.

Costantino, G., Matteucci, I., 2023. Reversing Kia motors head unit to discover and exploit software vulnerabilities. J. Comput. Virol. Hacking Tech. 19 (1), 33–49.

Cui, J., Liew, L.S., Sabaliauskaite, G., Zhou, F., 2019. A review on safety failures, security attacks, and available countermeasures for autonomous vehicles. Ad Hoc Netw. 90, 101823.

Ding, W., Alrashdi, I., Hawash, H., Abdel-Basset, M., 2024. DeepSecDrive: An explainable deep learning framework for real-time detection of cyberattack in in-vehicle networks. Inform. Sci. 658, 120057. http://dx.doi.org/10.1016/j.ins.2023.120057, URL: https://www.sciencedirect.com/science/article/pii/S0020025523016432.

El-Rewini, Z., Sadatsharan, K., Selvaraj, D.F., Plathottam, S.J., Ranganathan, P., 2020. Cybersecurity challenges in vehicular communications. Veh. Commun. 23, 100214. http://dx.doi.org/10.1016/j.vehcom.2019.100214.

ENISA, 2019. ENISA Good Practices for the Security of Smart Cars. Technical Report, European Union Agency for Cybersecurity (ENISA), Greece, URL: https://www.enisa.europa.eu/publications/smart-cars.

Fessi, B.A., BenAbdallah, S., Hamdi, M., Boudriga, N., 2009. A new genetic algorithm approach for intrusion response system in computer networks. In: 2009 IEEE Symposium on Computers and Communications. pp. 342–347. http://dx.doi.org/10.1109/ISCC.2009.5202379.

Fishburn, P.C., 1967. Additive utilities with incomplete product sets: Application to priorities and assignments. Oper. Res. 15 (3), 537–542.

Guo, Y., Zhang, H., Li, Z., Li, F., Fang, L., Yin, L., Cao, J., 2020. Decision-making for intrusion response: Which, where, in what order, and how long? In: ICC 2020 - 2020 IEEE International Conference on Communications. ICC, pp. 1–6. http://dx.doi.org/10.1109/ICC40277.2020.9149083.

Hamad, M., Hammadeh, Z.A., Saidi, S., Prevelakis, V., Ernst, R., 2018. Prediction of abnormal temporal behavior in real-time systems. In: Proceedings of the 33rd Annual ACM Symposium on Applied Computing. pp. 359–367.

Hamad, M., Nolte, M., Prevelakis, V., 2016. Towards comprehensive threat modeling for vehicles. In: The 1st Workshop on Security and Dependability of Critical Embedded Real-Time Systems.

Hamad, M., Prevelakis, V., 2020. SAVTA: A hybrid vehicular threat model: Overview and case study. Information 11 (5), http://dx.doi.org/10.3390/info11050273.

Hamad, M., Steinhorst, S., 2023. Security challenges in autonomous systems design. arXiv:2312.00018.

Hamad, M., Tsantekidis, M., Prevelakis, V., 2019. Red-Zone: Towards an intrusion response framework for intra-vehicle system. In: 5th International Conference on Vehicle Technology and Intelligent Transport Systems. VEHITS, http://dx.doi.org/10.5220/0007715201480158.

Hamad, M., Tsantekidis, M., Prevelakis, V., 2021. Intrusion response system for vehicles: Challenges and vision. In: Helfert, M., Klein, C., Donnellan, B., Gusikhin, O. (Eds.), Smart Cities, Green Technologies and Intelligent Transport Systems. Springer International Publishing, Cham, pp. 321–341.

Han, M.L., Kwak, B.I., Kim, H.K., 2021. Event-triggered interval-based anomaly detection and attack identification methods for an in-vehicle network. IEEE Trans. Inf. Forensics Secur. 16, 2941–2956. http://dx.doi.org/10.1109/TIFS.2021.3069171.

Heigl, M., Doerr, L., Almaini, A., Fiala, D., Schram, M., 2018. Incident reaction based on intrusion detections' alert analysis. In: 2018 International Conference on Applied Electronics. AE, pp. 1–6. http://dx.doi.org/10.23919/AE.2018.8501419.

Henniger, O., Ruddle, A., Seudié, H., Weyl, B., Wolf, M., Wollinger, T., 2009. Securing vehicular on-board it systems: The evita project. In: VDI/VW Automotive Security Conference. p. 41.

Herold, N., 2017. Incident Handling Systems with Automated Intrusion Response (Ph.D. thesis). Technische Universität München.

Herold, N., Posselt, S.-A., Hanka, O., Carle, G., 2016. Anomaly detection for SOME/IP using complex event processing. In: NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium. pp. 1221–1226. http://dx.doi.org/10.1109/NOMS.2016.7502991.

Herold, N., Wachs, M., Posselt, S.-A., Carle, G., 2017. An optimal metric-aware response selection strategy for intrusion response systems. In: Cuppens, F., Wang, L., Cuppens-Boulahia, N., Tawbi, N., Garcia-Alfaro, J. (Eds.), Foundations and Practice of Security. Springer International Publishing, Cham, pp. 68–84.

Hughes, K., McLaughlin, K., Sezer, S., 2020. Dynamic countermeasure knowledge for intrusion response systems. In: 2020 31st Irish Signals and Systems Conference. ISSC, pp. 1–6. http://dx.doi.org/10.1109/ISSC49989.2020.9180198.

Iannucci, S., Barba, O.D., Cardellini, V., Banicescu, I., 2019a. A performance evaluation of deep reinforcement learning for model-based intrusion response. In: 2019 IEEE 4th International Workshops on Foundations and Applications of Self* Systems (FAS*W). pp. 158–163. http://dx.doi.org/10.1109/FAS-W.2019.00047.

Iannucci, S., Casalicchio, E., Lucantonio, M., 2021. An intrusion response approach for elastic applications based on reinforcement learning. In: 2021 IEEE Symposium Series on Computational Intelligence. SSCI, pp. 01–10. http://dx.doi.org/10.1109/SSCI50451.2021.9659882.

Iannucci, S., Montemaggio, A., Williams, B., 2019b. Towards self-defense of non-stationary systems. In: 2019 International Conference on Computing, Networking and Communications. ICNC, pp. 250–254. http://dx.doi.org/10.1109/ICCNC.2019.8685487.

International Organization for Standardization, 2021. ISO/SAE 21434: 2021: Road Vehicles: Cybersecurity Engineering. ISO.

International Telecommunication Union, 2022. Guidelines for an intrusion prevention system for connected vehicles - recommendation ITU-T X.1377.

Islam, M.M., Lautenbach, A., Sandberg, C., Olovsson, T., 2016. A risk assessment framework for automotive embedded systems. In: Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security. pp. 3–14.

Jeong, S., Lee, S., Lee, H., Kim, H.K., 2024. X-CANIDS: Signal-aware explainable intrusion detection system for controller area network-based in-vehicle network. IEEE Trans. Veh. Technol. 73 (3), 3230–3246. http://dx.doi.org/10.1109/TVT.2023.3327275.

Jurewicz, C., Sobhani, A., Woolley, J., Dutschke, J., Corben, B., 2016. Exploration of vehicle impact speed – injury severity relationships for application in safer road design. Transp. Res. Procedia 14, 4247–4256. http://dx.doi.org/10.1016/j.trpro.2016.05.396, URL: https://www.sciencedirect.com/science/article/pii/S2352146516304021.

Karahasanovic, A., Kleberger, P., Almgren, M., 2017. Adapting threat modeling methods for the automotive industry. In: Ej Tryckt.

Kholidy, H.A., Erradi, A., Abdelwahed, S., Baiardi, F., 2016. A risk mitigation approach for autonomous cloud intrusion response system. Computing 98 (11), 1111–1135. http://dx.doi.org/10.1007/s00607-016-0495-8.

Kim, K., Kim, J.S., Jeong, S., Park, J.-H., Kim, H.K., 2021. Cybersecurity for autonomous vehicles: Review of attacks and defense. Comput. Secur. 103, 102150.

Klee, V., Minty, G.J., 1972. How good is the simplex algorithm? In: Inequalities III (Proc. Third Sympos., Univ. California, Los Angeles, Calif., 1969; Dedicated To the Memory of Theodore S. Motzkin). Academic Press, New York, pp. 159–175.

Kneib, M., Huth, C., 2018. Scission: Signal characteristic-based sender identification and intrusion detection in automotive networks. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. pp. 787–800.

Konak, A., Coit, D.W., Smith, A.E., 2006. Multi-objective optimization using genetic algorithms: A tutorial. Reliab. Eng. Syst. Saf. 91 (9), 992–1007.

Lautenbach, A., Almgren, M., Olovsson, T., 2021. Proposing HEAVENS 2.0–an automotive risk assessment model. In: Proceedings of the 5th ACM Computer Science in Cars Symposium. pp. 1–12.

Lokman, S.-F., Othman, A.T., Abu-Bakar, M.-H., 2019. Intrusion detection system for automotive controller area network (CAN) bus system: a review. EURASIP J. Wireless Commun. Networking 2019 (1), 184. http://dx.doi.org/10.1186/s13638-019-1484-3.

Lopes, A., Hutchison, A., 2020. Experimenting with machine learning in automated intrusion response. In: Kotenko, I., Badica, C., Desnitsky, V., El Baz, D., Ivanovic, M. (Eds.), Intelligent Distributed Computing XIII. Springer International Publishing, Cham, pp. 505–514.

Luo, F., Jiang, Y., Zhang, Z., Ren, Y., Hou, S., 2021. Threat analysis and risk assessment for connected vehicles: A survey. Secur. Commun. Netw. 2021, http://dx.doi.org/10.1155/2021/1263820.

Mahima, K.T.Y., Ayoob, M., Poravi, G., 2021. Adversarial attacks and defense technologies on autonomous vehicles: A review. Appl. Comput. Syst. 26 (2), 96–106. http://dx.doi.org/10.2478/acss-2021-0012.

Matthews, G., Feinstein, B., 2007. The Intrusion Detection Exchange Protocol (IDXP). RFC 4767, http://dx.doi.org/10.17487/RFC4767, URL: https://www.rfc-editor.org/info/rfc4767.

Miller, C., Valasek, C., 2015. Remote exploitation of an unaltered passenger vehicle. https://illmatics.com/Remote%20Car%20Hacking.pdf. (Accessed: 12 April 2022).

Mitchell, S., O'Sullivan, M., Dunning, I., 2011. PuLP: A Linear Programming Toolkit for Python. Department of Engineering Science, the University of Auckland, Auckland, New Zealand.

Nespoli, P., Papamartzivanos, D., Gómez Mármol, F., Kambourakis, G., 2018. Optimal countermeasures selection against cyber attacks: A comprehensive survey on reaction frameworks. IEEE Commun. Surv. Tutor. 20 (2), 1361–1396. http://dx.doi.org/10.1109/COMST.2017.2781126.

Olt, C., 2019. Establishing security operation centers for connected cars. ATZelectron. WorldWide 14 (5), 40–43.

Ossenbühl, S., Steinberger, J., Baier, H., 2015. Towards automated incident handling: How to select an appropriate response against a network-based attack? In: 2015 Ninth International Conference on IT Security Incident Management & IT Forensics. pp. 51–67. http://dx.doi.org/10.1109/IMF.2015.13.

Palanca, A., Evenchick, E., Maggi, F., Zanero, S., 2017. A stealth, selective, link-layer denial-of-service attack against automotive networks. In: Detection of Intrusions and Malware, and Vulnerability Assessment: 14th International Conference, DIMVA 2017, Bonn, Germany, July 6-7, 2017, Proceedings 14. Springer, pp. 185–206.

Papadaki, M., Furnell, S., Lines, B., Reynolds, P., 2003. Operational characteristics of an automated intrusion response system. In: Lioy, A., Mazzocchi, D. (Eds.), Communications and Multimedia Security. Advanced Techniques for Network and Data Protection. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 65–75.

Python Software Foundation, 2022. Resource — Resource usage information. https://docs.python.org/3/library/resource.html. (Accessed: 20 July 2022).

Richards, D.C., 2010. Relationship between speed and risk of fatal injury: pedestrians and car occupants. In: Road Safety Web Publication. Vol. 16, Department for Transport, London.

Rose, J.R., Swann, M., Grammatikakis, K.P., Koufos, I., Bendiab, G., Shiaeles, S., Kolokotronis, N., 2022. IDERES: Intrusion detection and response system using machine learning and attack graphs. J. Syst. Archit. 131, 102722.

Schrijver, A., 1998. The simplex method. In: Theory of Linear and Integer Programming. John Wiley & Sons, New York, pp. 129–150.

Sembera, V., 2020. ISO/SAE 21434: Setting the Standard for Connected Cars' Cybersecurity. White Paper, Trend Micro Research.

Shameli-Sendi, A., Ezzati-Jivan, N., Jabbarifar, M., Dagenais, M., 2012. Intrusion response systems: Survey and taxonomy. Int. J. Comput. Sci. Netw. Secur. (IJCSNS) 12.

Souissi, S., Serhrouchni, Sliman, L., Charroux, B., 2017. Security incident response: Towards a novel decision-making system. In: Madureira, A.M., Abraham, A., Gamboa, D., Novais, P. (Eds.), Intelligent Systems Design and Applications. Springer International Publishing.

Stakhanova, N., Basu, S., Wong, J., 2007. A taxonomy of intrusion response system. Int. J. Inf. Comput. Secur. 1, 169–184. http://dx.doi.org/10.1504/IJICS.2007.012248.

Stakhanova, N., Strasburg, C., Basu, S., Wong, J.S., 2012. Towards cost-sensitive assessment of intrusion response selection. J. Comput. Secur. 20 (2–3), 169–198.

Strasburg, C., Stakhanova, N., Basu, S., Wong, J.S., 2009. A framework for cost sensitive assessment of intrusion response selection. In: 2009 33rd Annual IEEE International Computer Software and Applications Conference. Vol. 1, IEEE, pp. 355–360.

Ullah, S., Khan, M.A., Ahmad, J., Jamal, S.S., e Huma, Z., Hassan, M.T., Pitropakis, N., Arshad, Buchanan, W.J., 2022. HDL-IDS: A hybrid deep learning architecture for intrusion detection in the internet of vehicles. Sensors 22 (4), http://dx.doi.org/10.3390/s22041340.

Ullah, S., Shelly, S., Hassanzadeh, A., Nayak, A., Hasan, K., 2020. On the effectiveness of intrusion response systems against persistent threats. In: 2020 International Conference on Computing, Networking and Communications. ICNC, pp. 415–421. http://dx.doi.org/10.1109/ICNC47757.2020.9049740.

Upstream, 2022. Upstream's 2022 global automotive cybersecurity report. URL: https://upstream.auto/2022report/.

Wang, B., Sun, Y., Sun, M., Xu, X., 2021a. Game-theoretic actor–critic-based intrusion response scheme (GTAC-IRS) for wireless SDN-based IoT networks. IEEE Internet Things J. 8 (3), 1830–1845. http://dx.doi.org/10.1109/JIOT.2020.3015042.

Wang, Y., Wang, Y., Qin, H., Ji, H., Zhang, Y., Wang, J., 2021b. A systematic risk assessment framework of automotive cybersecurity. Automot. Innov. 4 (3), 253–261. http://dx.doi.org/10.1007/s42154-021-00140-6.

Wolf, M., Weimerskirch, A., Paar, C., 2004. Security in automotive bus systems. In: Proceedings of the Workshop on Embedded Security in Cars (ESCAR)'04.

Wright, S., 2021. Autonomous Cars Generate More Than 300 TB of Data per Year. Tech Blog, Tuxera, Finland, URL: https://www.tuxera.com/blog/autonomous-cars-300-tb-of-data-per-year/.

Xia, S., Qiu, M., Liu, M., Zhong, M., Zhao, H., 2019. AI enhanced automatic response system for resisting network threats. In: Qiu, M. (Ed.), Smart Computing and Communication. Springer International Publishing, Cham, pp. 221–230.

Yarygina, T., Otterstad, C., 2018. A game of microservices: Automated intrusion response. In: Bonomi, S., Rivière, E. (Eds.), Distributed Applications and Interoperable Systems. Springer International Publishing, Cham, pp. 169–177.

Zonouz, S.A., Khurana, H., Sanders, W.H., Yardley, T.M., 2014. RRE: A game-theoretic intrusion response and recovery engine. IEEE Trans. Parallel Distrib. Syst. 25 (2), 395–406. http://dx.doi.org/10.1109/TPDS.2013.211.

Mohammad Hamad: He has been a research group leader with the Embedded Systems and Internet of Things Group at the Faculty of Computer Engineering, Technical University of Munich, Munich, Germany since 2020. He received his B.Eng. degree in Software Engineering and Information Systems from Aleppo University, Aleppo, Syria, in 2009. He also earned his Ph.D. (Dr.-Ing.) degree in Computer Engineering from the Institute for Data Technology and Communication Networks, Technical University of Braunschweig, Braunschweig, Germany, in 2020. His research interests lie in the area of autonomous vehicles and IoT security.

Andreas Finkenzeller: He received the B.Sc. and M.Sc. degrees in electrical engineering and computer science from Technical University Munich, Munich, Germany, in 2018 and 2021, respectively, where he is currently pursuing the Ph.D. degree with the Embedded Systems and Internet of Things Group. His research interests include embedded systems, secure communication, and IoT Security.

Michael Kühr: He received a B.Eng. degree in Electrical Engineering from the Baden-Wuerttemberg Cooperative State University in Stuttgart, Germany, in 2017 and a M.Sc. degree in Electrical Engineering and Information Technology from the Technical University of Munich, Munich, Germany, in 2022. His research interest focuses on the development and security of automated vehicles.

Andrew Roberts: He received the MCyberSecOps from the University of New South Wales, Canberra, Australia, in 2018 and the M.Sc. degree in cybersecurity engineering from Tallinn University of Technology in 2020. He is currently pursuing a Ph.D. degree in information technology at the Tallinn University of Technology, Estonia. His current research is focused on cybersecurity testing approaches to autonomous driving algorithms and methods to improve the robustness of the design of autonomous systems to cyber threats.

Olaf Maenne: He got his Ph.D. from the Technical University in Munich, studying wide-area Computer Networks and Network security through active and passive measurements and large-scale experiments. He has since then held faculty positions at Loughborough University in England and Tallinn University of Technology (TalTech) in Estonia, where he led the research at the Centre for Digital Forensics and Cybersecurity and established a Centre for Maritime Cybersecurity in Estonia. Since 2023, he has been with the University of Adelaide. His research interests have broadened over the years to include cyber defense technical exercises and critical infrastructure protection. He has been chairing numerous conferences, including ACM SIGCOMM in London in 2015 and the ACM Internet Measurement Conference (2017), and he is treasurer at ACM SIGCOMM 2024 in Sydney.

Vassilis Prevelakis: He received the B.Sc. degree (Hons.) in mathematics and computer science and the M.Sc. degree in computer science from the University of Kent, Canterbury, U.K., in 1984 and 1986, respectively, and the Ph.D. degree in computer science from the University of Geneva, Geneva, Switzerland, in 1996. He has worked in various areas of security in Systems and Networks both in his current academic capacity and as a freelance consultant. He is the Professor of Embedded Computer Security at the Technical University of Braunschweig, Braunschweig, Germany. His current research involves issues related to vehicular automation security, secure processors, security aspects of software engineering, and auto-configuration issues in secure VPNs.

Sebastian Steinhorst: He received the M.Sc. (Dipl.-Inf.) and Ph.D. (Dr. phil. nat.) degrees in computer science from Goethe University Frankfurt, Frankfurt, Germany, in 2005 and 2011, respectively. He is an Associate Professor at the Technical University of Munich, Munich, Germany, where he leads the Embedded Systems and Internet of Things Group, Department of Electrical and Computer Engineering. He was also a Co-Program PI in the Electrification Suite and Test Lab of the research center TUMCREATE in Singapore. Prof. Steinhorst's research centers around design methodology and hardware/software architecture co-design of secure distributed embedded systems for use in IoT, automotive, and smart energy applications.

Appendix VI

Paper VI

A. Roberts, M. R. H. Iman, M. Bellone, T. Ghasempouri, J. Raik, O. Maennel, M. Hamad, and S. Steinhorst. Adassure: Debugging methodology for autonomous driving control algorithms. In 2024 Design, Automation Test in Europe Conference Exhibition (DATE), pages 1–6, 2024.

2024 Design, Automation & Test in Europe Conference (DATE 2024)
Special Initiative "Autonomous Systems Design"

ADAssure: Debugging Methodology for Autonomous Driving Control Algorithms

Andrew Roberts[◇*], Mohammad Reza Heidari Iman[◇†], Mauro Bellone[*], Tara Ghasempouri[†],
Jaan Raik[†], Olaf Maennel[‡], Mohammad Hamad[§], Sebastian Steinhorst[§]
[*] FinEst Centre for Smart Cities, Tallinn University of Technology
[†] Department of Computer Systems, Tallinn University of Technology
[‡]School of Computer and Mathematical Sciences, The University of Adelaide
[§]Department of Computer Engineering, Technical University of Munich

***Abstract*—Autonomous driving (AD) system designers need methods to efficiently debug vulnerabilities found in control algorithms. Existing methods lack alignment to the requirements of AD control designers to provide an analysis of the parameters of the AD system and how they are affected by cyber-attacks. We introduce ADAssure, a methodology for debugging AD control system algorithms that incorporates automated mechanisms which support generation of assertions to guide the AD system designer to identify vulnerabilities in the system. Our evaluation of ADAssure on a real-world AD vehicular system using diverse cyber-attacks developed a set of assertions that identified weaknesses in the OpenPlanner 2.5 AD planning algorithm and its constituent planning functions. Working with an AD control system designer and safety validation engineer, the results of ADAssure identified remediation of the AD control system, which can support the implementation of a redundant observer for data integrity checking and improvements to the planning algorithm. The adoption of ADAssure improves autonomous system design by providing a systematic approach to enhance safety and reliability through the identification and mitigation of vulnerabilities from corner cases.**

***Index Terms*—Security, Autonomous Driving**

I. Introduction

Autonomous driving (AD) vehicles are increasingly being utilised for transportation on public roads. Waymo and Cruise offer AD ride-hailing services in San Francisco, Apollo Baidu in China, and numerous such services are operating in Europe. Central to the wider-adoption of AD vehicles on public roads is the security and safety of their control algorithms that enable self-driving technology. AD control algorithms comprise a complex code-base of interconnected modules that perform tasks and sub-tasks that enable a vehicle to sense, perceive, localise, and navigate in a driving environment. With the increase in diversity of AD use-cases from valet parking to public transportation in public traffic, the code base of AD control algorithms will reputedly grow from 100-200 million to billions of lines of code [1].

Within this complex environment, debugging the code for logical errors arising from unexpected control behaviour is a fundamental challenge [2]. AD system designers need to pinpoint where in the control software weaknesses are, in order to focus debugging efforts in an efficient manner. Existing studies attempt to rectify unexpected AD control behaviour at run-time through smoothing trajectories utilising neural networks [3] [4] [5]. The applicability of these studies in real-world AD programs are limited due to the highly dynamic environment of autonomous driving and the probabilistic nature of the algorithms for planning.

[◇] These authors contributed equally to this work.

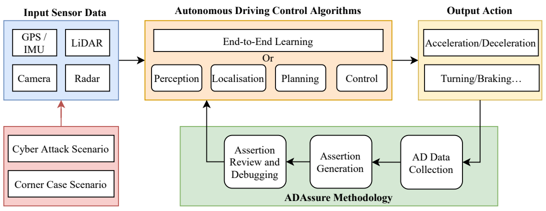


Fig. 1: Comprehensive ADAssure methodology overview that illustrates each step of the process, from data collection to assertion creation, review of assertions, and debugging.

Furthermore, in these studies, the analysis lacks the expertise from the algorithm designer and safety engineer to inform on the nature of the behaviour of vehicle dynamics, whether noise identified as irregular could be considered for a control engineer within normal constraints, whether AD behaviour could be considered a legitimate safety response to an unexpected event and whether the parameters for which the run-time solution is designed are appropriate for differing class of vehicles with different dynamic profiles. We consider the design phase to offer the most promising area of initial investigation to improve the robustness of control algorithms, which can be translated to real-world AD systems.

In this work, we propose ADAssure, a methodology for debugging control algorithms during the design-time phase of AD control software development (Fig. 1). ADAssure is built upon the idea that the data of vehicle dynamics and sensing of AD systems can be analysed for anomalous control behaviour, which can then be transformed into assertions on the AD control. We use association rules that enable us to mine datasets of varying scales and fingerprint the pattern of anomalous activity. These rules can be used to guide AD system designers to focus on the debugging of the control algorithms. To evaluate ADAssure, we focus on a control system algorithm used in a real-world AD vehicular system providing ride-hailing services. To summarise, the paper makes the following contributions:

- We propose ADAssure, a methodology designed for debugging AD control algorithms during the autonomous system development's design phase (see § II).
- To demonstrate our methodology's feasibility, we applied it to diverse datasets, revealing three new vulnerabilities in `Openplanner 2.5` AD, used in real-world vehicles. (see § IV).
- We provide to the community our artifacts to enable reproducibility and assist with developing efforts to im-

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on June 13,2025 at 15:59:48 UTC from IEEE Xplore. Restrictions apply.

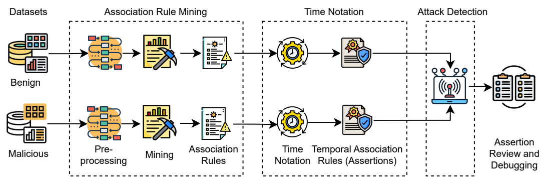


Fig. 2: Phases for Assertion Generation

prove AD control system design. These artifacts include simulation datasets and real-world AD system data (*ADAssure Datasets*).

II. ADAssure: Methodology

The development of ADAssure has three main motivations. First, it aims to provide AD system designers with a methodology to identify and fix vulnerabilities that align with the design of AD algorithms. Second, given the ever-changing nature of the autonomous vehicle system, it strives to establish a structured methodology that allows for consistent, flexible, and repeatable testing. Third, it aims to support unit testing, allowing testing of individual components of the autonomous system in isolation from other dynamic factors affecting autonomous control.

The foundations of the ADAssure methodology are based on the analysis of the vehicle dynamics and sensing data to guide the creation of assertions of the vulnerability of the AD control algorithms. The analysis consists of a sensitivity analysis of vehicle dynamics data (e.g., velocity, yaw, and steering angle), sensor data (e.g., lateral and longitudinal movement), and visualisation of the trajectory of the AD system. This helps identify key parameters to build assertions of the AD control algorithms. The AD control system designers can use the assertions to identify and locate the vulnerabilities of the control model and develop mechanisms to test and fix the errors. The ADAssure methodology comprises three main phases: AD Data Collection, Association Rule Generation, and Assertion Review and Debugging. Next, we will explore each phase in more depth.

A. Autonomous Driving Data Collection

This phase consists of generating data from the real-world system or simulation environment. The benefit of a simulation environment is that driving scenarios can be automated or designed to test a specific condition, such as a cyber-attack or a corner case. The data output is structured according to established metrics. These can be vehicle dynamics parameters (yaw angle, velocity, etc.), sensing data (position co-variance, point-cloud, etc.), and safety parameters (distance-to-collision, etc.). The AD data is outputted in a format that can be interpreted by analytical tools, in our use-case, .csv format.

B. Association Rule Generation Phase

The goal of this phase is to process the data generated from the previous phase and produce a set of association rules that can be translated into assertions in the Assertion Review and Debugging phase. This phase is comprised of three primary steps (as shown in Fig. 2): a) Association Rule Mining, b) Time Notation, and c) Attack Detection. The association rule mining is applied to both benign and malicious datasets, resulting in two distinct sets of association rules.

Algorithm 1: Association rule mining & time notation

```
1  Input: $\mathcal{N}$, $\mathcal{D}$
2  Output: $next[\mathcal{N}] = antecedent \rightarrow next[\mathcal{N}]consequent$,
   $before[\mathcal{N}] = antecedent \rightarrow before[\mathcal{N}]consequent$
   /* Initialization and Preprocessing */
3  $\mathcal{R}$ = $antecedent \rightarrow consequent$
4  forall $f \in \mathcal{D}$ do
5      $\mathcal{D}'$ = MoveUp($f(\mathcal{N})$)
   /* Mining */
6  $\mathcal{R} \leftarrow$ apriori($\mathcal{D}'$)
   /* Time Notation */
7  if $(\mathcal{R}.antecedent == (t \in \mathcal{D}'))$ and $(\mathcal{R}.consequent == (f \in \mathcal{D}'))$ then
8      $next[\mathcal{N}] \leftarrow$ label($\mathcal{R}$)
9  if $(\mathcal{R}.antecedent == (f \in \mathcal{D}'))$ and $(\mathcal{R}.consequent == (t \in \mathcal{D}'))$ then
10     $before[\mathcal{N}] \leftarrow$ label($\mathcal{R}$)
```

These rules are then processed through the Time Notation step to incorporate temporal information, yielding temporal association rules (assertions) in the form of $next[\mathcal{N}]$ and $before[\mathcal{N}]$ patterns. We define $next[\mathcal{N}]$ type of rule in the general form of $\mathcal{X} \rightarrow next[\mathcal{N}]\mathcal{Y}$. This rule indicates that when $\mathcal{X}$ occurs, after $\mathcal{N}$ time instants, $\mathcal{Y}$ will occur. $\mathcal{N}$ is a positive integer value. Moreover, we define $before[\mathcal{N}]$ rule in the general form of $\mathcal{X} \rightarrow before[\mathcal{N}]\mathcal{Y}$. This rule demonstrates that whenever $\mathcal{X}$ happens, $\mathcal{Y}$ should have occurred $\mathcal{N}$ time instants before that. The "Attack Detection" step compares these temporal association rules, ultimately detecting attacks and anomalies within the datasets. Subsequent sections provide a more in-depth discussion of each step.

a) Association Rule Mining: This step primarily serves two objectives: pre-processing the datasets and subsequently mining association rules from the preprocessed data. To mine the association rules, apriori algorithm [6] was adopted and enhanced to mine temporal rules capable of detecting attacks at various time instances during autonomous vehicle (AV) operation. Algorithm 1 presents the details of the Association Rule Mining and Time Notation steps. In this algorithm, $\mathcal{D}$ denotes the dataset and $\mathcal{D}'$ is the preprocessed dataset, while f and t represent the dataset's features and target values. To prepare the dataset for mining the $next[\mathcal{N}]$ and $before[\mathcal{N}]$ temporal patterns, all the features of the dataset are moved $\mathcal{N}$ records above its original position (Line 5). However, the target of the dataset remains as it is. Afterwards, the apriori algorithm is applied to the preprocessed dataset to mine a set of association rules. The output of this phase is a set of association rules in the general form of $antecedent \rightarrow consequent$ that are ready to be forwarded to the Time Notation step.

b) Time Notation: In this step, the method integrates the concept of time into the association rules generated in the association rule mining step, leading to a set of temporal association rules. The method determines to which temporal pattern ($next[\mathcal{N}]$ or $before[\mathcal{N}]$) each extracted rule belongs and subsequently assigns the corresponding time label to the rule. If the antecedent value matches a target value in the dataset, and the consequent value has already been moved to another record in the dataset, the rule is labelled as a $next$ temporal association rule (Line 8). Otherwise, if the antecedent of a rule mined in the association rule mining step matches a dataset feature that has already been moved

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on June 13,2025 at 15:59:48 UTC from IEEE Xplore. Restrictions apply.

to another record and the consequent of the rule matches the target value of the dataset, we label this rule as a $before$ temporal association rule (Line 10). The mined rules are in the forms of $antecedent \rightarrow next[\mathcal{N}]consequent$, and $antecedent \rightarrow before[\mathcal{N}]consequent$, serving as assertions for debugging the AD system.

c) Attack Detection: This step aims to identify rules indicating attacks on the AV. We assume that the sets of mined rules from the benign and malicious datasets should be similar under normal conditions, without any AV attacks. Any deviation between these rule sets signifies an anomaly in the autonomous vehicle. Per this assumption, the temporal association rules (assertions) mined during the time notation phase are classified into two sets. The first category comprises rules exclusively mined from the malicious dataset, lacking counterparts in the benign dataset. Any rule extracted solely from the malicious dataset, without a corresponding counterpart in the benign dataset, signifies an attack. These rules reveal abnormal behaviour in the malicious dataset, contrasting with different behaviour observed in the corresponding time instance of the benign dataset. Consequently, we classify these as attacks. The second category comprises similar rules mined from both benign and malicious datasets, but with different *minimum support* (`min_supp`) and *minimum confidence* (`min_conf`) values. The variations in these values indicate that, while the mined rules are similar, abnormal behaviours and anomalies exist between the datasets. The apriori algorithm employs these two metrics (i.e., `min_supp` and `min_conf`). The `min_supp` value is the threshold and a minimum value that is chosen by the expert to decide whether a rule occurs frequently in the dataset or not [7], [8]. The `min_conf` is the minimum value that is chosen by the expert and is an indication of how often a rule has been found to be true [6], [9]. Increasing the `min_supp` value results in fewer association rules that describe more general behaviour of the autonomous vehicle, while decreasing the `min_supp` value leads to rules covering rare behaviours (corner cases). Similarly, raising the `min_conf` value produces fewer but more valid rules. Valid rules refer to association rules that will not be violated with different attack scenarios like corner cases. These values in the ADAssure facilitate an effective attack detection process. The second category of rules aids the ADAssure in effectively identifying corner cases and the attacks that rarely occur on the AV. These rare attacks exhibit behaviour very similar to normal vehicle operation but are malicious and can lead to AV failure.

C. Assertion Review and Debugging

Within this phase, the association rules generated from the association rule mining are reviewed in conjunction with an analysis of the control behaviour and individual data parameters to develop assertions. Trajectory maps of the AD system and graphs, which demonstrate the sensitivity of the data parameters during benign and cyber-attack scenarios, are compared to the anomalous behavioural patterns detected by the association rule mining tool. Using expertise from the algorithm designer and safety validation engineer assists in understanding which parameters can uniquely demonstrate a vulnerability of an algorithm within the system. From developing an assertion on the system's vulnerability, the debugging effort focuses on a control flow analysis. As the assertion assists in pinpointing the specific module, the static analysis can focus on the control flow of the substituent functions within the module. As an example of the importance of this pinpointing, a local-planning module could have 15 diverse algorithms, and within these, each could have multiple different methods or functions. As the code of AD algorithms are differential equations, debugging can suggest optimisations that enable mitigation mechanisms against the identified vulnerabilities.

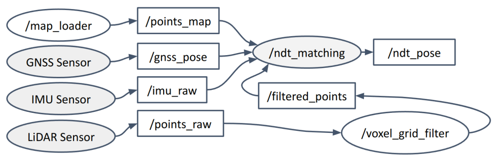


Fig. 3: Localisation Algorithm Flow within AD System.

III. Autonomous Driving Control Algorithm

To evaluate the methodology, we focus on an AD control algorithm used in a real-world AD ride-hailing service. Within the AD pipeline, there are four key modules: localisation, perception, planning, and control. Within our study, we focus on the localisation and planning modules.

A. Localisation Module

This module provides accurate information regarding the position and orientation of the vehicle. Using a Normal Distributions Transform (NDT) matching search algorithm, it identifies the best matching position based on sensor perception. It uses input from the Inertial measurement unit (IMU) and the point cloud generated by the LiDAR. Then, it attempts to match the points from our current scan to a grid of probability functions derived from the map. NDT matching algorithms can also benefit from the GNSS sensor, which provides initial rough estimates of localization on geo-referenced maps, thereby avoiding any sudden errors in localization calculations that may result in failures. Fig. 3 displays the flow of the localisation algorithm within the AD system.

B. Planning Module

For the AD system to plan a mission, firstly, a global planner generates a global reference path using a vector (road network) map. The function of the global planner is to stipulate a route between the starting position and goal position of the mission on the road map. The local-planner generates smooth and obstacle-free trajectories in the operational local domain following the global route. The local-planner consists of several modules (see Fig. 4); trajectory generation, trajectory evaluation, intention and trajectory estimator, object-tracker and behavior selection (decision making) [10]. The trajectory generation module generates alternative tracks parallel to the main path defined by the global planner. These tracks are named roll-outs. The trajectory evaluation module assesses all possible roll-outs and the data input from sensed-data of the AV and makes a cost estimation. The behaviour selector will lead the AV to motion on a roll-out based on the least-cost.

IV. Experimentation and Results

To evaluate the impact of corner cases on AD system behaviour using the ADAssure methodology, we use datasets of corner cases from simulation and real-world driving from

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on June 13,2025 at 15:59:48 UTC from IEEE Xplore. Restrictions apply.

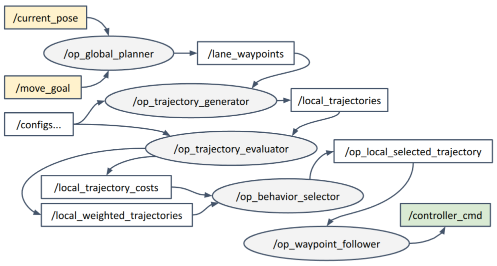


Fig. 4: Abstract Local Planning Algorithm Flow within AD System.

the target AD system. The 1st corner case scenario dataset is of three diverse cyber-security attacks on the AD system conducted in a simulation environment. As our focus is the planning and localisation algorithms, we used a low-fidelity simulation provided by `Autoware.AI` and the `OpenPlanner 2.5` planning algorithm. The 2nd corner case scenario dataset is of a Global Positioning System (GPS) spoofing event that occurred on the AD system during its operation on the roads of a capital city.

A. *AD Control System Datasets*

a) Cyber-security Corner Case Dataset: Within this dataset, three attacks were conducted on the target AD vehicular system, which is attempting an overtaking manoeuvre. The three attacks are classified as: 1) *Lateral Position Offset Attack* 2) *Longitudinal Position Offset Attack* 3) *Message Time-Delay*. In the lateral and longitudinal position offset attack, an attacker injects malicious data input into the lateral or longitudinal pose whilst the AD vehicular system is in the process of the overtaking manoeuvre (Fig. 5). This attack could be conducted through GPS spoofing or interception and manipulation of the localisation sensor data. The attacker introduces a delay into the `current_pose` (lateral and longitudinal) sensor messages reaching the AD control pipeline for the message time-delay. The malicious data is injected at around the $21\,\mathrm{m}$ mark of the AV journey (travelled distanced) to the $67\,\mathrm{m}$. Each attack was conducted 300 times, accommodating a variation of different attack parameters. The lateral and longitudinal attacks introduced a deviation ranging from $0.16\,\%$ to $1.0\,\%$, which equates to around $20\,\mathrm{cm}$ to $1\,\mathrm{m}$. The message time-delay introduced delays of $0.3\,\%$, $0.6\,\%$, $1.0\,\%$ second, as a message is transmitted every $20\,\mathrm{ms}$, this range represents a delay of $15\,\mathrm{to}\ 50\,\mathrm{messages}$. In total, the dataset comprises over $1500\,\mathrm{scenario}$ runs of attacks and benign safety cases.

b) GPS Spoofing Real-World AV Dataset: The AD ride-hailing service transmits its sensor data via a logging node to an edge server, which stores the AD System data in a database.

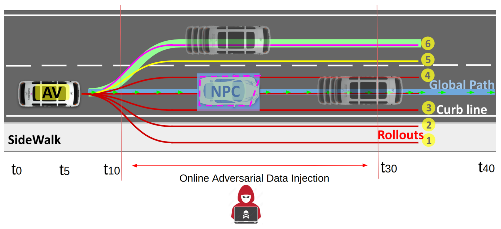


Fig. 5: The threat model used for conducting the attack cases.

TABLE I: AD System Data.

AD Data Type	Description
AV_X	Longitudinal Position of the AD System as to the HD Map
AV_Y	Lateral Position of the AD System as to the HD Map
AV_Steer	Steering Angle of the AD System
AV_Vel	Velocity of the AD System
AV_Yaw	Orientation of the AD System based on its centre of gravity
Roll-out_Num	Current Lane according to the lane selector of the AD Control Algorithm
DTC	Distance to collision of the AD vehicular system to the overtaking vehicle.
Position Co-variance	GPS position co-variance
Altitude	Altitude derived from the GPS

During its operations near the port area of the city, the AD vehicle encountered a loss of localisation from a GPS spoofing event which also affected other GPS-enabled platforms. This GPS spoofing continued intermittently throughout the preceding months. The dataset used in this study is from the logging system of AD ride-hailing service.

c) AD System Data: The simulation and real-world datasets were structured to output data as shown in Table I.

B. *Experimental Results*

To evaluate the ADAssure methodology, we chose six attack types and their corresponding safety (benign) scenarios. These attack types included each of the aforementioned attacks with differing levels of noise (lateral and longitudinal position offset, delay message).

a) Automated Analysis: Utilising the ADAssure methodology on the three types of attacks yields three distinct set of assertions corresponding to each attack type. The results of the assertion generation phase are presented in Table II. The threshold for minimum support (`min_supp`) is set at 0.01 , while the minimum confidence (`min_conf`) threshold is 1 Notably, the method exhibits a swift execution time. Within the 3 attacks of the cybersecurity corner case dataset, the assertions identify two patterns of anomalous AD behaviour. Firstly, extreme steering angles of $20°$ and $-20°$ and sudden lane transition. Secondly, multiple lane-transitions combined with the extreme steering angle and sudden changes in vehicular velocity. This behaviour can be seen to be the effect of cyber activity on the smoothness of the initiation of the overtaking manoeuvre which results in turbulent movements and in some cases, a collision event. The assertions generated from the GNSS spoofing dataset identified the changes to the altitude and position co-variance. These were consistent with dramatic change in the values of the GPS coordinates and the resultant change in altitude.

b) Assertion Review and Debugging: The patterns identified in the association rules enables us to extrapolate that the Yaw angle and angular velocity are good reference point to show the effect of cyber-attacks. During the injection of the position offset attacks, the vehicle's orientation demonstrates dramatic action; in some circumstances, the vehicle can be

TABLE II: ADAssure Assertion Generation phase results.

Dataset		Assertion			Execution Time
Name	#Records	Total	$\#Next[\mathcal{N}]$	$\#Before[\mathcal{N}]$	
Longitude	412	5	3	2	1 ns
Latitude	356	7	7	0	1 ns
Delay	417	5	3	2	1 ns
GNSS	16	5	4	1	1 ns

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on June 13,2025 at 15:59:48 UTC from IEEE Xplore. Restrictions apply.

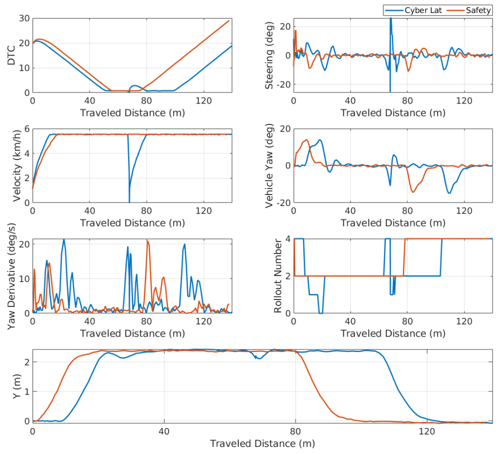


Fig. 6: Lateral position offset attack vehicle parameters.

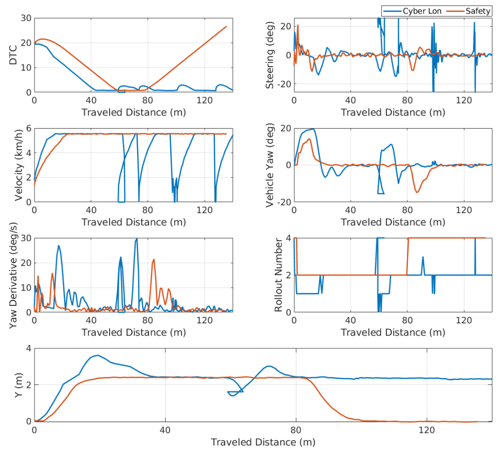


Fig. 7: Longitudinal position offset attack vehicle parameters.

seen to be essentially spinning. As displayed in Fig. 6, the Lateral Position Offset Attack displays the Yaw (angle) of the vehicle making sharp changes, of 15 deg/sec from 15 meters mark of the AV journey. This vehicle dynamic behaviour is a characteristic also seen in both the longitudinal position offset (Fig. 7) and delay message attack (Fig. 8). The results for the velocity parameter demonstrate that it only indicates immediate collision of the vehicle, and it does not support early identification of anomalous vehicle behaviour. Assertion 1 contends that the AD system should not allow movements that challenge the physical limitations of the steering model.

Assertion 1: To determine the vulnerability of the yaw angle and momentum, we can derive the assertion: *AV.displacement_of_yaw_angle* $>$ *max_yaw_angle_threshold && time* $<$ *time_threshold*.

The roll-out transition, steer, and distance-to-collision parameters demonstrate identifiable change during a cyber-

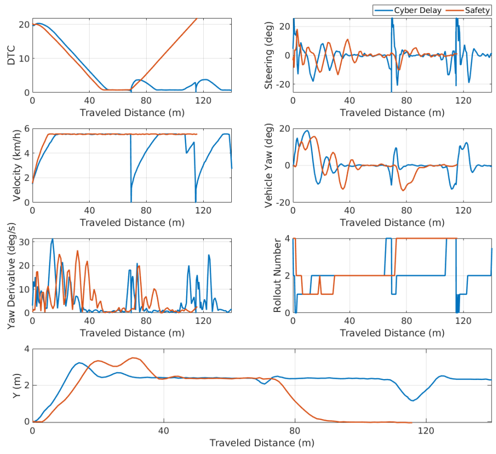


Fig. 8: Delay message attack vehicle parameters.

attack. The manipulation of the lateral and longitudinal position alters the vehicle position on the map and, therefore, has the effect of inducing greater transitions between roll-outs, which is the effective position of the vehicle on the road. The frequency of transition impacts the smoothness of the steering angle. From the distance-to-collision parameter, it is noted that the effect of the attack is most prominent during the overtaking maneuver and mostly during the cut-in process, when the vehicle cuts-in front of the passing vehicle (NPC). Assertion 2 contends that when the vehicle transitions across multiple roll-outs and displays 180° steering and closes to less than $0.5\,\mathrm{m}$ to the passing vehicle, this represents affected behaviour from the cyber attack.

Assertion 2: To identify vehicle dynamic changes from cyber-attack: *AV.x* $-$ *NPC.x* $<$ *distance_threshold && AV.lane_transition* $>$ *max_transition_number &&* *AV.steer_angle* $\notin$ *[min, max]_steer_angle*

Assertion 3 contends with activity seen in the longitudinal position offset (Fig. 7) where the AV collides with the passing vehicle and then accelerates to the previous set-point.

Assertion 3: To identify collisions we can derive the assertion: $|AV.v_k - AV.v_{k+1}| > threshold.$

Assertion 3 could also be used to detect anomalies in GPS data. The GNSS spoofing attack demonstrates a significant deviation in the altitude and position co-variance parameters. Assuming that velocity data comes from two sources, a wheel sensor measurement and calculated by deriving the position from GPS data, the two results should be close to each other. In the case of a GNSS spoofing attack, the deviation in the position co-variance would generate a spike in the velocity (calculated by deriving the position in GPS data), and thus violating assertion 3.

For our specific AD system, the threshold for assertion 1 is 15° yaw angle displacement within $1\,\mathrm{s}$ duration. Assertion 2 threshold is identified as a distance between AV and passing

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on June 13,2025 at 15:59:48 UTC from IEEE Xplore. Restrictions apply.

vehicle as less than $0.5\,\mathrm{m}$, lane transition greater than 1 roll-out and steering angle that is outside the bounds of 20 and $-20°$. It is important to note that these values are valid for a low-speed AV ride-hailing service and for designers of different classes of vehicles, it is required to calculate values consistent with their specific application.

Solvable bugs come from several points in the controller; a simple one is wrong or imprecise saturation values of the control signal, which generates a high acceleration or a high steering angle in the vehicle. This is clearly visible in Fig. 7 where a signal overshoot causes the vehicle to change lane multiple times. Another example, clearly visible in Fig. 6,7& 8 is the lack of a fallback plan. There is a clear indication of a collision as the vehicle speed suddenly drops to $0\,\mathrm{m\,s^{-1}}$ and then quickly accelerates to the reference point, this is a violation of Assertion 3. A robust controller should have a fallback plan for such a case which indicates a bug in the functional design of the controller. In such a case, the vehicle should be aware of the fact that the global trajectory cannot be followed anymore and switch to emergency mode.

The main reason for searching for unexpected behaviours is to debug the controller, with reference to the experimental results, a violation of Assertion 1 can be associated to a bug in the `/ndt_pose` module (see Fig. 3), while a violation of Assertion 2 can be backpropagated to the module `/op_trajectory_evaluator`. A violation of assertion three can be backpropagated to the modules of `/op_trajectory_generator` and `/op_behaviour_selector` (see Fig. 3). To pinpoint the violation of assertion 3 to a specific function, we abstracted from the local_planner algorithm and its substituent lane_rule algorithm, the `getClosestWaypointNumber` method, which selects the next waypoint to follow in the global trajectory and returned an exception to be handled as a different driving behaviour (e.g., there was a crash, emergency mode activated).

In the case of GNSS attack, the NDT localisation algorithm doesn't detect the deviation in position co-variance, and this is due to the normal vector pointing in the same direction. Debugging focuses on optimisation of the NDT localisation using visual odometry for holding the local position at short-distances until the source of the disturbance has been resolved.

V. Related Work

Recent publications on anomaly detection in vehicular AD control systems propose the usage of vehicle dynamics as a key detection indicator for cyber-attacks [11] [12] [13]. Studies such as Guo et al. [14] emphasise the effect cyber-attacks have on the trajectory of the AD system and the noise of individual sensors. Mitigation mechanisms focus on two diverse approaches 1) implementation of an observer of AD vehicle state estimation which can inform an emergency action (sensor switching etc.) [14] 2) implementation of trajectory smoothing algorithm to correct unplanned vehicle behaviour [12] [13]. However, these solutions for detection and mitigation are developed based on assumptions of driving environment and algorithm configuration and this limits the scope of their applicability.

VI. Conclusion

Cyber-attacks present new challenges to the design of AD algorithms. Designers need methods to debug vulnerabilities to improve robustness. In this paper, we introduced ADAssure, a methodology for debugging AD algorithms during the design phase. The methodology consisted of three phases; 1) AD Data collection 2) Assertion Rule Generation 3) Assertion Review and Debugging. The concept of the methodology was to develop association rules from mining AD data which can be transformed into assertions on the vulnerability of the system.

Our evaluation of ADAssure on diverse cyber-security datasets from simulation and real-world revealed that the ADAssure method could identify three assertions on the vulnerability of the `OpenPlanner 2.5` AD planning algorithm. These assertions were able to guide an algorithm designer and safety engineer to pinpoint the specific modules in the planning algorithm for debugging.

Acknowledgment

This work has been supported by the European Commission through the European Union's Horizon 2020 Research and Innovation Programme, under grant agreement No 101021727.

References

[1] Bosch, "Facts and figures about electronics and software in vehicles," *Automotive World*, July, 2021.

[2] W. Zeng, M. Wu, P. Chen, Z. Cao, and S. Xie, "Review of shared online hailing and autonomous taxi services," *Transportmetrica B: Transport Dynamics*, vol. 11, no. 1, pp. 486–509, 2023.

[3] K. K.-C. Chang, X. Liu, C.-W. Lin, C. Huang, and Q. Zhu, "A safety-guaranteed framework for neural-network-based planners in connected vehicles under communication disturbance," in *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2023.

[4] R. Jiao, H. Liang, T. Sato, J. Shen, Q. A. Chen, and Q. Zhu, "End-to-end uncertainty-based mitigation of adversarial attacks to automated lane centering," in *2021 IEEE Intelligent Vehicles Symposium (IV)*, 2021.

[5] X. Liu, R. Jiao, B. Zheng, D. Liang, and Q. Zhu, "Safety-driven interactive planning for neural network-based lane changing," in *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, ser. ASPDAC '23. Association for Computing Machinery, 2023.

[6] J. Han, M. Kamber, and J. Pei, "6 - mining frequent patterns, associations, and correlations: Basic concepts and methods," in *Data Mining (Third Edition)*, ser. The Morgan Kaufmann Series in Data Management Systems. Boston: Morgan Kaufmann, 2012, pp. 243–278.

[7] M. Zaki, "Scalable algorithms for association mining," *IEEE Transactions on Knowledge and Data Engineering*, 2000.

[8] M. R. Heidari Iman, J. Raik, M. Jenihhin, G. Jervan, and T. Ghasempouri, "An automated method for mining high-quality assertion sets," *Microprocessors and Microsystems*, vol. 97, p. 104773, 2023.

[9] M. Shahin, M. R. Heidari Iman, M. Kaushik, R. Sharma, T. Ghasempouri, and D. Draheim, "Exploring factors in a crossroad dataset using cluster-based association rule mining," *Procedia Computer Science*, 2022.

[10] H. Darweesh, E. Takeuchi, and K. Takeda, "Openplanner 2.0: The portable open source planner for autonomous driving applications," in *2021 IEEE Intelligent Vehicles Symposium Workshops*, 2021.

[11] 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.

[12] Y. Ma, J. Sharp, R. Wang, E. Fernandes, and X. Zhu, "Sequential attacks on kalman filter-based forward collision warning systems," in *AAAI Conference on Artificial Intelligence*, 2020.

[13] J. Shen, Y. Luo, Z. Wan, and Q. A. Chen, "Lateral-direction localization attack in high-level autonomous driving: Domain-specific defense opportunity via lane detection," 2023.

[14] J. Guo, L. Li, J. Wang, and K. Li, "Cyber-physical system-based path tracking control of autonomous vehicles under cyber-attacks," *IEEE Transactions on Industrial Informatics*, 2023.

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on June 13,2025 at 15:59:48 UTC from IEEE Xplore. Restrictions apply.

Appendix VII

Paper VII

A. Roberts, M. Malayjerdi, M. Bellone, O. Maennel, and E. Malayjerdi. Analysing adversarial threats to rule-based local-planning algorithms for autonomous driving. Inaugural Symposium on Vehicle Security and Privacy (VehicleSec 2023) with NDSS, pages 1–8, 2023.

Analysing Adversarial Threats to Rule-Based Local-Planning Algorithms for Autonomous Driving

Andrew Roberts
FinEst Centre for Smart Cities
Tallinn University of Technology
Andrew.Roberts@taltech.ee

Mohsen Malayjerdi
Department of Mechanical and Industrial Engineering
Tallinn University of Technology
Mohsen.Malayjerdi@taltech.ee

Mauro Bellone
FinEst Centre for Smart Cities
Tallinn University of Technology
mauro.bellone@taltech.ee

Olaf Maennel
School of Computer and Mathematical Sciences
The University of Adelaide
olaf.maennel@adelaide.edu.au

Ehsan Malayjerdi
Department of Mechanical and Industrial Engineering
Tallinn University of Technology
ehsan.malayjerdi@taltech.ee

***Abstract*—The safety and security of navigation and planning algorithms are essential for the adoption of autonomous driving in real-world operational environments. Adversarial threats to local-planning algorithms are a developing field. Attacks have primarily been targeted at trajectory prediction algorithms which are used by the autonomous vehicle to predict the motion of ego vehicles and other environmental objects to calculate a safe planning route. This work extends the attack surface to focus on a rule-based local-planning algorithm, specifically focusing on the planning cost-based function, which is used to estimate the safest and most efficient route. Targeting this algorithm, which is used in a real-world, operational autonomous vehicle program, we devise two attacks; 1) deviation to the lateral and longitudinal pose values, and 2) time-delay of the sensed-data input messages to the local-planning nodes. Using a low-fidelity simulation testing environment, we conduct a sensitivity analysis using multiple deviation range values and time-delay duration. We find that the impact of adversarial attack cases is visible in the rate of failure to complete the mission and in the occurrence of safety violations. The cost-function is sensitive to deviations in lateral and longitudinal pose and higher duration of message delay. The result of the sensitivity analysis suggests minor deviations of the pose (lateral, longitudinal) values as an optimal range for the attackers search space. Options for mitigating such attacks are that the AV should run a concurrent process executing a concurrent planning instance for redundancy.**

I. Introduction

Navigation and planning algorithms are essential for autonomous driving (AD). For the self-driving vehicle to navigate the road environment, the navigation and path-planning algorithm must calculate a route that ensures safety for the passenger and external environmental actors (pedestrians, other vehicles and road users, etc.) and achievement of the journey (mission). Initial studies of navigation and path planning algorithms for AD have shown them to be vulnerable to adversarial attacks that introduce uncertainties into the route calculation, which causes downstream effects on the safe behavioural control of the AV [2], [17], [3]. To improve the reliability of navigation and planning algorithms, they need to be further tested for uncertainties, and these methods are incorporated into the architecture of autonomous driving.

There are a few studies that focus on adversarial attacks on local-planning. These studies target machine learning algorithms for local-planning modules such as trajectory prediction (Trajectron++, Agentformer and GRIP++) [2], [17], [3]. The predominant threat model adopted, focuses on developing methods and tools of adversarial learning to understand the trajectory prediction model of the target AV and then either crafting malicious sensor data input or training other ego AVs in the driving environment to interfere with the target AVs predicted trajectory [15], [2], [17], [3]. The required result of a successful adversarial attack is to cause the target AV to generate a trajectory that is unsafe, inefficient, or uncomfortable for passengers. In this work, we expand on the target of attacks to a rule-based algorithm for local-planning, and focus on the trajectory generation and estimation of an AV. Our justification for focusing on rule-based algorithms is that, whilst AI approximate reasoning algorithms seem to be highly promising for the near future, an impediment to current adoption is the lack of feedback in real-world driving scenarios [5]. Rule-based algorithms for path-planning in robot navigation and AD are well-established, and more ubiquitous in real-world deployments.

A rule-based local-planning algorithm uses a cost function to estimate the least-cost path. The cost function takes input from immediately sensed-data; current pose, velocity etc.. The cost estimation is based on a calculation of factors such as; lateral collision, longitudinal collision, lane transition, central deviation etc., and weighting is given to these factors based on criteria such as safety and efficiency. By interpreting the cost-function, used for trajectory generation and estimation, as part of local-planning, an adversarial attack can be crafted which affects the downstream behavioural control whose decisions impact the safe driving state of the AV.

The main idea of this paper is that the white-box knowledge of the cost estimation function of the rule-based local planning

Symposium on Vehicles Security and Privacy (VehicleSec) 2023
27 February 2023, San Diego, CA, USA
ISBN 1-891562-88-6
https://dx.doi.org/10.14722/vehiclesec.2023.23086
www.ndss-symposium.org

algorithm can be used to craft adversarial attacks by manipulating factors inherent to the cost function. Evaluating white-box generated attacks enable an understanding of the level of stealth of the adversarial threat, and whether adversarial manipulation by the cyber attack can be distinguished from noise. Furthermore, these attacks will enable evaluation and assessment of the optimisation of the algorithm to uncertainties and the quality of decision-making.

The key questions this study engages are the following:

1) What is the sensitivity of the cost function to adversarial data manipulation of key driving parameters?
2) How can an adversarial attack hide in the cost function from detection?
3) What optimisations of the rule-based algorithm can be considered to mitigate against adversarial data manipulation?

The problem area of this research, is centred on a local-planning algorithm, open-planner 2.5, which is used in an AV shuttle program that operates in real-world road conditions in Europe [7]. As with the open-source software community, development of vulnerability research and testing methods proliferate across the ecosystem and are utilised and innovated for diverse platforms. The aim of this study is to focus on the vulnerability of the local-planning function of autonomous driving and provide direction and guidance to the autonomous driving security community to develop vulnerability testing on diverse planners and algorithms. In a broader sense, this research aims to understand how AD algorithms used in real-world AD programs can be tested for adversarial threats and validated to improve assurance for real-world operational driving.

II. Rule-Based Local-Planning Algorithm

A. *Open Planner 2.5 Local-Planner Overview*

For the AV to plan a mission, firstly, a global planner generates a global reference path using a vector (road network) map. The function of the global planner is to stipulate the starting position and goal position of the mission on the road map. How to achieve this mission, for the AV to navigate from the starting position to the mission goal, through a smooth, obstacle free trajectory is the function of the local-planner. The local-planner consists of several modules (see Figure 1); trajectory generation, trajectory evaluation, intention and trajectory estimator, object-tracker and behavior selection (decision making) [7]. The trajectory generation module generates alternative tracks parallel to the main path defined by the global planner. These tracks are named rollouts. The trajectory evaluation module assesses all possible rollouts and the data input from sensed-data of the AV and makes a cost estimation. The behaviour selector will lead the AV to motion on a rollout based on the least-cost.

Table I displays the input and outputs of each of the local-planning modules (Note. intention and trajectory estimator and object-tracker are not visible as they are not within the scope of this study).

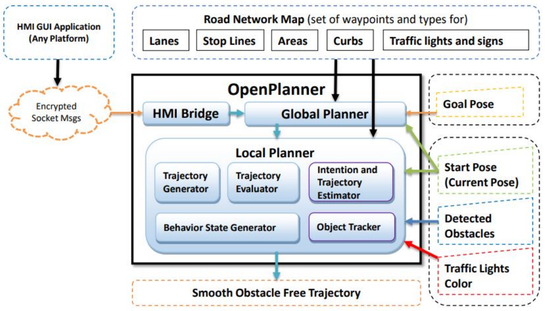


Fig. 1: OpenPlanner 2.5 Architecture [7]

TABLE I: Local-Planning Module

Node	**Input**	**Output**
Trajectory Generator	Initial_Pose	Local Trajectories
	Current_Pose	
	Current_Velocity	
	Lane_Waypoints_Array	
Trajectory Evaluator	Current_Velocity	Local Trajectory_Cost
	Current_Pose	
	Local_Trajectories	
	Lane_Waypoints_Array	
	Predicted_Objects	
	Current_Global_Local_IDS	
Behavioural Selector	Current_Velocity	Current_Behaviour
	Current_Pose	
	Local_Trajectory_Cost	
	Local_Weighted_Trajectories	

B. *Local Planning Cost Function*

The local motion planning algorithm generates a trajectory (or a set of control commands for the AV) by minimizing a cost function, within a workspace, that includes a set of design parameters. The cost function constitutes the rules for motion-planning which inform the decision-making for autonomous driving.

The cost function is built on five factors and calculated in the following Eq. 1:

$$C = \begin{bmatrix} w_{cent} \\ w_{trans} \\ w_{longColl} \\ w_{latColl} \\ w_{vis} \end{bmatrix} \cdot \begin{bmatrix} C_{cent} \\ C_{trans} \\ C_{longColl} \\ C_{latColl} \\ C_{vis} \end{bmatrix}^T \tag{1}$$

where, C_{cent} is the cost associated to the central trajectory and is designed to keep the vehicle in the central trajectory; C_{trans} is the transition cost that prevents the vehicle from jumping between rollouts; $C_{longColl}$ and $C_{latColl}$ are the cost of the longitudinal and lateral collision respectively, and finally C_{vis} is the weight associated to the visibility [12]. Each of these costs are weighted by their respective weighting factors w_i [6].

III. Threat Model

The attack targets the local planning cost function, with the aim of inducing the trajectory evaluation to choose a motion-planning route that is not optimal for safety, functionality of the driving mission and comfort of the passengers. To achieve this, the most direct mechanism to impact the cost function is to manipulate, with adversarial data, the sensed-data input that is inherent to local-planning. The Current_Pose data is the optimal target for this as it is the primary sensed-data for localisation of the vehicle, containing the longitudinal, lateral positioning and orientation of the AV. Whilst altering the pose data of the vehicle has previously been conducted in other studies [2], [17], [3], [15], in our attack we aim to explore the sensitivity of our cost function to data manipulations and conducting the attack during specific time-intervals.

For the threat model used in our study, we assume that the attacker has access to the internal network of the AV and is able to listen to control message communications and collect data. This could be achieved through supply-chain compromise of a library in the control software, insider threat actor, or many of the vulnerabilities in existing communication frameworks for autonomous systems such as the robotic operating system (ROS) [8]. Given the attacker has access to the internal network, the question arises, why not change the Lane_ID or a driving parameter which would be more simplistic and direct? We view these attacks as overt in nature and likely to be detected, the compelling nature of adversarial data manipulation is that the attack is difficult for AV safety engineers to interpret between noise and an explicit cyber threat. Another consideration are the external interfaces of the vehicle localisation sensing, which generates the pose data. It is a possibility that the pose data can be manipulated by an external attack in the form of GPS spoofing or an adversarial LiDAR, dependent on the sensor configuration used for the localisation of the vehicle. The study focused on the vulnerability of the planner and its search space, considering localisation. We considered internal attacks to be important due to the increase in attacks through software and hardware supply-chains, and therefore the scope of the attacks within the study highlighted this area.

A. *Attack Case 1: Position Offset Attack*

The attacker creates a spoofed ROS topic which is able to deliver malicious input data of the Current_Pose (longitude, latitude, and velocity) to all the nodes of the local planning module. The data manipulation is injected online/dynamically during the critical overtaking manoeuvre involving the AV and NPC (Non-playable character). Figure 2 displays the critical driving scenario and the time frames in which the manipulated Current_Pose data is injected into the local planning pipeline cost estimation. The red dashed lines in Figure 2 represent the roll-outs, and the green highlighted, denoting the selected motion-path.

For the manipulation of the Current_Pose data, we introduce a deviation to lateral and longitudinal pose.

For the lateral pose data, the sensitivity deviation introduced was structured as follows:

- Attack Case 1a: 0.16%

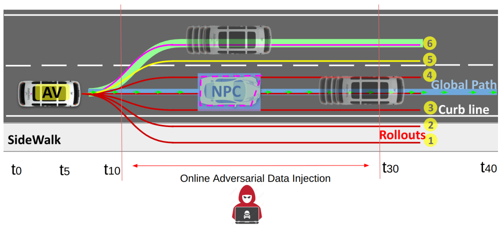


Fig. 2: Threat Model

- Attack Case 1b: 0.33%
- Attack Case 1c: 0.5%

In designing the range of deviation, we considered state-of-the-art attacks such as AdvDO attack [2], which noted two requirements for developing adversarial threats to planning algorithms:

1) Malicious data input needs to be feasible to the real, physical constraints of the vehicle [2].
2) Malicious data input of the local-planning algorithm should be close to the nominal trajectory [2].

Therefore, we chose a range from a slight perturbation of pose to a 1m deviation.

The longitudinal pose data sensitivity deviation range was structured as follows:

- Attack Case 1d: 0.33%
- Attack Case 1e: 0.66%
- Attack Case 1f: 1.00%

This range is the same as the longitudinal deviation. The difference in percentage comes from the difference in coordinate values of lateral and longitude. The lateral value is almost double those of the longitudinal, and therefore the percentage is doubled.

B. *Attack Case 2: Message Time-Delay*

For the second attack case, we inserted a time-delay into the messages of the Current_Pose topic communicating to the nodes of the local planning module.

We introduced a message delay when the AV passes 2m in front of the NPC (from the centre) in the lateral direction. We introduce 3 different time delays in the message:

- Attack Case 2a: 0.3 seconds
- Attack Case 2b: 0.6 seconds
- Attack Case 2c: 1.0 seconds

The message frequency is approximately 50hz, so this is a message every 20 milliseconds. We chose the above range of deviation of time-delay as it enabled a spectrum of a message from the delay from approximately 15, to 50 messages.

IV. Experimental Setup

A. Test Environment and Configuration

In terms of conducting such experiments, simulation is the best method among all testing methods for AVs. To accelerate the testing, we bypassed the sensing and detection nodes of the algorithm and focused on the planning part by utilizing the low-fidelity simulation feature provided by Autoware.ai and Openplanner. The low-fidelity simulation uses the openplanner 2.5 control algorithm. It provides simulated localization and detection data for the planning nodes and receives the actuation commands to simulate the AV kinematics. This process runs faster due to the low-detail environment required for the simulation and the lack of the process to simulate the sensors. Figure 3 displays the different frames of an overtaking simulation in the simulator.

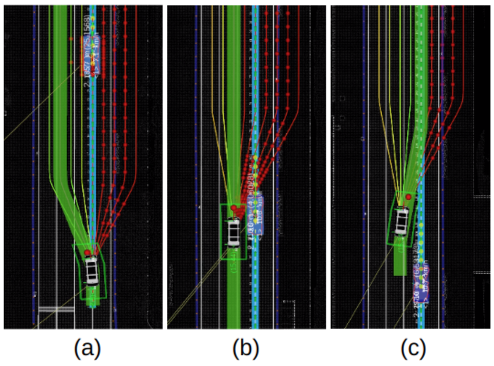


Fig. 3: Example of an overtaking simulation in the low-fidelity simulator, a) starting point of the overtaking b) middle of the mission, AV is on the opposite lane reaching the NPC c) AV cuts in

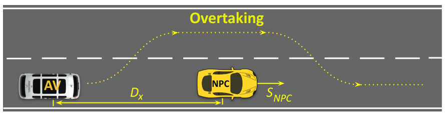


Fig. 4: Target scenario, Dx and S_{NPC}, define the initial relative distance to the NPC and the constant NPC speed in the scenario

1) Target Mission: Overtaking is one of the most challenging maneuvers for Avs [10]. In this research, we selected this operation as the target scenario for studying the planning algorithm under the cyber-attack. The scenario parameters in Figure 4 are listed in Table II.

TABLE II: Target scenarios definition

Actor	Speed (m/s)	$D_x(m)$	Goal
AV	[0:6]	0	overtake the NPC safely
NPC	3	25	keep moving

2) Safety Evaluation Test: To assess the safety and reliability of the planning algorithm in normal conditions (no attack), we ran the scenario simulation 300 times to reach a meaningful statistical population. Then, the planning algorithm behavior in each case was evaluated with the local-planner performance evaluation criteria (explained in the next section).

3) Attack Test Cases: Finally, the platform was used to simulate the proposed adversarial data manipulations and time-delay messaging, during the overtaking mission and monitor the algorithm's behavior. For each attack case, we ran the simulation (with attack) 100 times. Overall, 900 simulations were conducted for all attack cases.

B. Evaluation Criteria

For the evaluation, we used previously established safety criterion [11] with evaluation criteria recommended by SafeBench, a benchmarking framework for safety evaluation of AD algorithms for critical driving scenarios [16]. Figure III displays the metrics used for the performance evaluation.

TABLE III: Local-Planner Performance Evaluation Criteria

Condition	**Data Label**	**Description**	**Metric**
Safety Violation	**V**		
Succeed	Suce	AV Successful complete the mission	Pass/Fail
Not Finished	NotF	Failure to finish the mission	Pass/Fail
Distance-to -Collision	DTC	Violation of the safe distance between AV and NPC	AV within 0.5m of other vehicle
Break on Driving Lane	BrD	AV initiates emergency break on driving lane	Pass/Fail
Break on Passing Lane	BrP	AV initiates emergency break on passing lane	Pass/Fail
Collision	Col	AV collides with NPC	Pass/Fail
Functionality			
Avg. time spent to complete route	TS	The average time taken to complete the mission	seconds
Comfort			
Avg. Acceleration	ACC	Average acceleration of the AV	m/s
Avg. Steering Angle	YV	Steering angle of the AV	degrees
Freq. of Lane Invasion	LI	The number of times the AV transitions to another rollout	numeric

V. Results

After running 1200 simulations, all recorded data including the AV and the NPC position and orientation were processed to assess the simulations based on the evaluation criteria. We also visualized the recorded data to study the violation and their cause in each simulation as shown in Figure 5. Figure 5.a represents a safety run completed successfully. Next, (b) and (c) display lateral and longitudinal attack cases which experienced brake and collision safety violations respectively. Finally, (d) shows a message time delay attack which is finished by a collision. The asterisk signs in the AV trajectory show the point where the Openplanner changes the rollout. Overall, all the safety violation results for the whole experiment are presented in Figure 6.

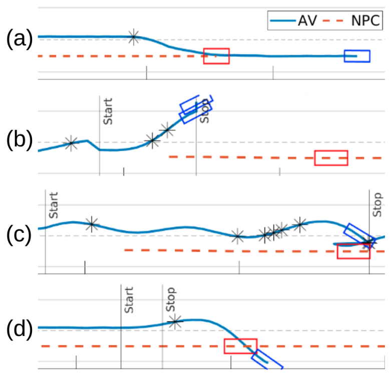


Fig. 5: 2D representation of the simulation of each test group. a) a successful safety test, b) a lateral attack case that led to a brake violation, c) a longitudinal attack case that experienced a collision, and d) a message time delay that causes a collision. for the attack cases a vertical line shows the start and stop point of the attack

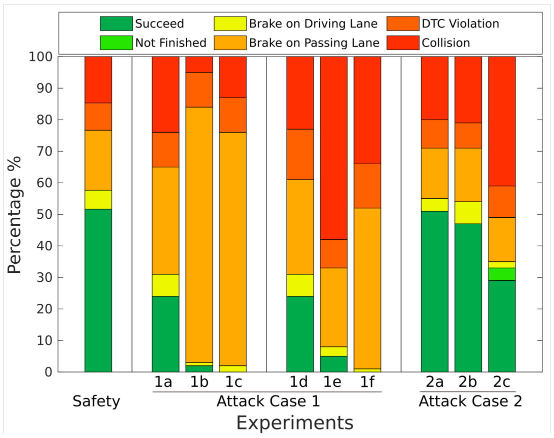


Fig. 6: All simulation result based on the proposed safety criteria

For each of the attack test cases, we saw an increase in safety violations of the AV compared to the normal safety test case experiment. As the value of the deviation for lateral and longitudinal values increased the number of successful mission completions decreased. Although marginal, the greater number of safety violations for the attacks on the Current_Pose data were observed in the lateral deviations. Given the importance of lateral positioning to the overtaking manoeuvre, this can be understood as any deviation increases the complexity of executing the overtaking manoeuvre. In the 1f attack test case, the highest value longitudinal change (approximately 1 meter) led to a crash with curbside and not able to continue the mission. This event was reported as a braking safety violation.

The time-delay messaging attack test case saw the only result for mission not finished metric. Furthermore, the greater the delay of the Current_Pose data reaching the local-planning nodes, the increased likelihood that a safety violation will occur, and in the case of our experiments, the greater the likelihood of a the most serious safety violation, collision.

Table IV demonstrates the results of the safety test according to the performance evaluation criteria. The level of safety violations are reflective of an algorithm which is in development and being optimised for critical driving scenarios such as overtaking.

TABLE IV: Summary of the Safety Simulation

$Num.$	V_{Col}	V_{DTC}	V_{BrP}	V_{BrD}	V_{NotF}	V_{Suce}
300	4.6%	8.6%	19%	6%	0%	51.6%

	TS	ACC	YV	LI
mean	29.1	0.4	3.8	7.1
STD	6.7	0.2	2.2	4.6
min	21.9	0.2	1.8	2
max	42.3	1.3	21.7	25

Table V shows that for each deviation there is a high number of safety violations in comparison to the safety test case results. In regards to the sensitivity analysis, a smaller deviation of around 20 to 25 cm can achieve the result that the local-planning algorithm is only successful in generating a trajectory that completes the mission in 24% of the total test set. Furthermore, a small deviation in the lateral pose, can achieve a higher number of collisions with an ego vehicle. It may also be seen from the lane invasion and steering angle results that small deviations to lateral pose result in a fluctuation of the cost of different rollouts which cause greater lane transitions as the cost function causes the AV to choose a route based on minimum cost. The higher deviation results in a higher occurrence of breaking activity and hitting the curb. Furthermore, the higher deviation results in the AV being stuck in the passing lane, this is due the dramatic change in lateral pose. The 1 meter deviation attack case results in 0% success of finishing the mission.

Table VI results of the longitudinal deviations also display a high number of safety violations in comparison to the safety test case results. Collision safety violation is highest for the longitudinal deviation attack. This can be reasoned as the longitudinal deviation does not experience the same high volume of breaking passing lane safety violations, where the vehicle gets stuck, as seen with the lateral pose deviation. The higher deviation of longitudinal pose, results in increased acceleration and this causes sharp breaking. This is indicated with the 1f result, the 1 meter deviation attack case, which displays a higher instance of breaking safety violation. The 1 meter deviation attack case results in 0% success of finishing the mission.

Table VII demonstrates the shorter delay of local pose data has minimal impact on the success of the mission and safety violations. As the time duration of the message delay is increased the impact to the reliability of the local-planning algorithm is higher. Test 2c, which is the delay of Current_Pose data of 1.0 second, shows considerable increases in collisions and decreases in the likelihood of the success of the mission.

TABLE V: Summary of the Attack Case 1: Position Offset Attack Simulation

Case	$Num.$	V_{Col}	V_{DTC}	V_{BrP}	V_{BrD}	V_{NotF}	V_{Suce}
1a	100	24%	11%	34%	7%	0%	24%
1b	100	5%	11%	81%	1%	0%	2%
1c	100	13%	11%	74%	2%	0%	0%

1a		TS	ACC	YV	LI
	mean	35.3	0.4	9	7.5
	STD	7.4	0.2	7.5	5.4
	min	21.9	0.2	1.9	1
	max	42.4	1	23	23

1b		TS	ACC	YV	LI
	mean	41.4	0.4	9.5	4.8
	STD	3.5	0.1	4.4	3
	min	22.1	0.2	3.1	1
	max	42.4	1.2	23.7	21

1c		TS	ACC	YV	LI
	mean	41.7	0.4	7.8	4.7
	STD	1.7	0.1	1.2	2.7
	min	32	0.3	4.3	1
	max	42.3	1	9.8	15

TABLE VI: Summary of the Attack Case 1: Position Offset Longitudinal Deviation Simulation

Case	$Num.$	V_{Col}	V_{DTC}	V_{BrP}	V_{BrD}	V_{NotF}	V_{Suce}
1d	100	23%	16%	30%	7%	0%	24%
1e	100	58%	9%	25%	3%	0%	5%
1f	100	34%	14%	51%	1%	0%	0%

1d		TS	ACC	YV	LI
	mean	33.8	0.5	5.7	9.1
	STD	7.6	0.3	4.9	5.4
	min	18.1	0.2	1.7	2
	max	43.2	1.4	23	27

1e		TS	ACC	YV	LI
	mean	32.2	0.6	6.7	10.5
	STD	9.5	0.2	3.2	5
	min	17.8	0.2	1.9	2
	max	43.2	1.1	20.5	25

1f		TS	ACC	YV	LI
	mean	32.2	0.7	5.9	11.3
	STD	7.9	0.2	2.5	4.7
	min	18	0.3	2.7	2
	max	43.2	1.4	22.1	26

The time-delay of the pose data to the local-planning nodes results in a loss of localisation and the greater delay the greater impact on the cost calculation which in turn causes uncertainty for the behaviour selector/decision-making.

TABLE VII: Summary of the Attack Case 2: White-Box Delay Simulation

Case	$Num.$	V_{Col}	V_{DTC}	V_{BrP}	V_{BrD}	V_{NotF}	V_{Suce}
2a	100	20%	9%	16%	4%	0%	51%
2b	100	21%	8%	17%	7%	0%	47%
2c	100	41%	10%	14%	2%	4%	29%

2a		TS	ACC	YV	LI
	mean	29.3	0.4	4.2	7.6
	STD	8.1	0.2	2.2	5.4
	min	18.1	0.2	1.8	2
	max	53	1.1	16.7	24

2b		TS	ACC	YV	LI
	mean	30.6	0.4	4.8	7.8
	STD	8.6	0.3	3.7	4.8
	min	22.9	0.2	1.8	2
	max	58	1.1	23.8	21

2c		TS	ACC	YV	LI
	mean	32.9	0.4	7	8.3
	STD	9.6	0.3	5.2	5
	min	13	0.2	1.1	0
	max	58.2	1.3	22.9	23

VI. Discussion

The results of the test simulations demonstrated that the cost function is sensitive to minor deviations of both the lateral and longitudinal pose. The success rate of the mission is visibly diminished when adding adversarial data manipulations to the sensed-data input. The higher the deviation, the higher the likelihood of mission failure. The minor deviation attacks, where the deviation is a range of 20 to 25cm offer a good starting point to mutate adversarial data for further attacks based on this range. Whilst the higher range attacks conducted in our experiments showed a higher rate of mission failure, a deviation of 1 meter can be seen a noisy enough to be observable. We also noticed such behaviour in a real-world AV shuttle [14] and a manual emergency break had to be enacted to prevent an emergency.

The time-delay attack demonstrated that minor delays cause minimal impact on the success of the mission and the occurrence of safety violations. Delays in sensed-data input flowing to the local-planning modules of greater than 1 second increase the rate of mission failure and safety violations. Given that 1 message is broadcast every 20 milliseconds, 1 second represents around 50 messages, and a delay of this magnitude is also likely to be more observable.

For the attack to hide in the cost function, investigating mutations for minor deviations of lateral and longitudinal values in the range of 20 to 30 cm, offer an optimal target range.

Mitigation of the adversarial deviation and time-delay attack could include the implementation of a redundant driver. This means that the AV should run a concurrent process executing a concurrent planning instance. If the redundant driver and the actual driving algorithm give different results, then this could indicate that an attack might be happening. In such a case, the AV could either stop safely awaiting for human intervention or switch to the redundant driver to complete its mission. The development of the architecture for a redundant driving integrity checking function also needs to consider isolation from the primary driving function so that an attacker cannot also compromise both.

VII. Related Work

As safety validation of AD algorithms is a critical field for the adoption of AD in real-world environments, there is a focus on testing the reliability of trajectory prediction and generation to adversarial driving actors in the road environment. Wang et al. [15], Abeysirigoonawardena, Dudek & Shkurti [1], Chen et al. [4], Klischat et al. [9], and O'Kelly et al. [13] use simulation environments to develop adversarial trained NPCs whose driving actions cause safety violations of the trajectory prediction of the targeted AV. These simulations are focused on safety validation and are not focused on the exploitation of the algorithm by adversarial threat actors, however, their methods in generating adversarial examples and target parameters and data values are of great use in developing adversarial cyber threats.

On a practical level, involving the real-world operation of AVs, there are few research studies into the robustness of planning and navigation algorithms to adversarial threats. Prominent among them are Zhang et al. [17], Cao et al. [3] and Cao et al. [2]. These studies focus on the robustness of the trajectory prediction, the ability of the AV to predict the trajectory of another ego vehicle or environmental object (pedestrian, animals etc.) and make driving decisions accordingly. The attacks in these studies are targeted at deep-neural networks (DNNs), and therefore focus on adversarial learning to develop robust adversarial trajectories. In relation to our work, the observations on ranges for deviation of lateral and longitudinal values and the considerations for crafting adversarial data were useful in developing our attack cases.

VIII. Conclusion

In this work, we conducted a sensitivity analysis of the openplanner 2.5 rule-based planning algorithm to adversarial data manipulation of lateral and longitude values and delayed sensed-input messages to local-planning nodes. We evaluated these attacks in a low-fidelity simulation test environment using an overtaking manoeuvre critical driving scenario. The results showed that the planning cost-function is sensitive to adversarial data manipulation that introduces deviations to the lateral and longitudinal values. These adversarial deviations cause higher rates of failure to complete missions and cause safety violations. For the message delay attack, limited delays in the range up to approximately 0.6 seconds have a limited impact on the trajectory calculation. Message delays for 1 second or greater cause a visible difference in the safety violation rate and mission success. We opine that limited deviations are an optimal area to explore further attacks and in more diverse critical driving scenarios.

Through this work we proposed a class of stealthy attacks on the local-planning function of AD. An area of future research is the development of monitoring systems developed around such basis of attacks. The results show the feasibility of monitoring real-time properties of the messages propagations and therefore post-mortem forensics might be able to determine the presence of an attacker causing safety violations of AVs.

IX. Acknowledgements

This work has been supported by the European Commission through the H2020 teaming project Finest Twins (grant No. 856602) and European Union's Horizon 2020 Research and Innovation Programme, under grant agreement No 101021727.

References

[1] Y. Abeysirigoonawardena, F. Shkurti, and G. Dudek, "Generating adversarial driving scenarios in high-fidelity simulators," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE Press, 2019, p. 8271–8277. [Online]. Available: https://doi.org/10.1109/ICRA.2019.8793740

[2] Y. Cao, C. Xiao, A. Anandkumar, D. Xu, and M. Pavone, "Advdo: Realistic adversarial attacks for trajectory prediction," in *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part V*. Berlin, Heidelberg: Springer-Verlag, 2022, p. 36–52. [Online]. Available: https://doi.org/10.1007/978-3-031-20065-6_3

[3] Y. Cao, D. Xu, X. Weng, Z. Mao, A. Anandkumar, C. Xiao, and M. Pavone. [Online]. Available: https://arxiv.org/abs/2208.00094

[4] B. Chen, X. Chen, Q. Wu, and L. Li, "Adversarial evaluation of autonomous vehicles in lane-change scenarios," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, pp. 10 333–10 342, 2020.

[5] L. Claussmann, M. Revilloud, D. Gruyer, and S. Glaser, "A review of motion planning for highway autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 5, pp. 1826–1848, 2020.

[6] H. Darweesh, E. Takeuchi, K. Takeda, Y. Ninomiya, A. Sujiwo, L. Y. M. Saiki, N. Akai, T. Tomizawa, and S. Kato, "Open source integrated planner for autonomous navigation in highly dynamic environments," *J. Robotics Mechatronics*, vol. 29, pp. 668–684, 2017.

[7] H. Darweesh, E. Takeuchi, and K. Takeda, "Openplanner 2.0: The portable open source planner for autonomous driving applications," in *2021 IEEE Intelligent Vehicles Symposium Workshops (IV Workshops)*, 2021, pp. 313–318.

[8] G. Deng, G. Xu, Y. Zhou, T. Zhang, and Y. Liu, "On the (in)security of secure ros2," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 739–753. [Online]. Available: https://doi.org/10.1145/3548606.3560681

[9] M. Klischat and M. Althoff, "Generating critical test scenarios for automated vehicles with evolutionary algorithms," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 2352–2358.

[10] E. Malayjerdi, R. Sell, M. Malayjerdi, A. Udal, and M. Bellone, "Practical path planning techniques in overtaking for autonomous shuttles," *Journal of Field Robotics*, vol. 39, no. 4, pp. 410–425, 2022.

[11] M. Malayjerdi, A. Roberts, O. m. Maennel, and E. Malayjerdi, "Combined safety and cybersecurity testing methodology for autonomous driving algorithms," in *Proceedings of the 6th ACM Computer Science in Cars Symposium*, ser. CSCS '22. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: https://doi.org/10.1145/3568160.3570235

[12] P. Narksri, H. Darweesh, E. Takeuchi, Y. Ninomiya, and K. Takeda, "Occlusion-aware motion planning with visibility maximization via active lateral position adjustment," *IEEE Access*, vol. 10, pp. 57 759–57 782, 2022.

[13] M. O'Kelly, A. Sinha, H. Namkoong, J. Duchi, and R. Tedrake, "Scalable end-to-end autonomous vehicle testing via rare-event simulation," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., 2018, p. 9849–9860.

[14] R. Sell, M. Leier, A. Rassõlkin, and J.-P. Ernits, "Self-driving car iseauto for research and education," in *2018 19th International Conference on Research and Education in Mechatronics (REM)*, 2018, pp. 111–116.

[15] J. Wang, A. Pun, J. Tu, S. Manivasagam, A. Sadat, S. Casas, M. Ren, and R. Urtasun, "Advsim: Generating safety-critical scenarios for self-driving vehicles," *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[16] C. Xu, W. Ding, W. Lyu, Z. Liu, S. Wang, Y. He, H. Hu, D. Zhao, and B. Li, "Safebench: A benchmarking platform for safety evaluation of autonomous vehicles," 2022. [Online]. Available: https://arxiv.org/abs/2206.09682

[17] Q. Zhang, S. Hu, J. Sun, Q. A. Chen, and Z. M. Mao, "On adversarial robustness of trajectory prediction for autonomous vehicles," *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*. [Online]. Available: https://par.nsf.gov/biblio/10359466

Appendix VIII

Paper VIII

A. Roberts, S. Marksteiner, M. Soyturk, B. Yaman, and Y. Yang. A global survey of standardization and industry practices of automotive cybersecurity validation and verification testing processes and tools. SAE International Journal of Connected and Automated Vehicles, 7, 11 2023

Downloaded from SAE International by Tallinn University of Technology, Friday, June 13, 2025

ARTICLE INFO
Article ID: 12-07-02-0013
© 2024 International Alliance for Mobility Testing and Standardization (IAMTS)
doi:10.4271/12-07-02-0013

A Global Survey of Standardization and Industry Practices of Automotive Cybersecurity Validation and Verification Testing Processes and Tools

Andrew Roberts,[1] Stefan Marksteiner,[2,6] Mujdat Soyturk,[3] Berkay Yaman,[4] and Yi Yang[5]

[1]*Tallinn University of Technology, Estonia*
[2]*AVL List GmbH, Austria*
[3]*Marmara Üniversitesi, Turkey*
[4]*BigTRI, Turkey*
[5]*AVL China, China*
[6]*Mälardalen University, Sweden*

Abstract

The United Nation Economic Commission for Europe (UNECE) Regulation 155—Cybersecurity and Cybersecurity Management System (UN R155) mandates the development of cybersecurity management systems (CSMS) as part of a vehicle's lifecycle. An inherent component of the CSMS is cybersecurity risk management and assessment. Validation and verification testing is a key activity for measuring the effectiveness of risk management, and it is mandated by UN R155 for type approval. Due to the focus of R155 and its suggested implementation guideline, ISO/SAE 21434:2021—Road Vehicle Cybersecurity Engineering, mainly centering on the alignment of cybersecurity risk management to the vehicle development lifecycle, there is a gap in knowledge of proscribed activities for validation and verification testing. This research provides guidance on automotive cybersecurity testing and verification by providing an overview of the state-of-the-art in relevant automotive standards, outlining their transposition into national regulation and the currently used processes and tools in the automotive industry. Through engagement with state-of-the-art literature and workshops and surveys with industry groups, our study found that national regulatory authorities are moving to enshrine UN R155 as part of their vehicle regulations, with differences of implementation based on regulatory culture and pre-existing approaches to vehicle regulation. Validation and verification testing is developing aligned to UN R155 and ISO21434:2021; however, the testing approaches currently used within industry utilize elements of traditional enterprise information technology methods for penetration testing and toolsets. Electrical/electronic (E/E) components such as embedded control units (ECUs) are considered the primary testing target; however, connected and autonomous vehicle technologies are increasingly attracting more focus for testing.

© 2024 International Alliance for Mobility Testing and Standardization (IAMTS). Published by SAE International. This Open Access article is published under the terms of the Creative Commons Attribution Non-Commercial, No Derivatives License (http://creativecommons.org/licenses/by-nc-nd/4.0/), which permits use, distribution, and reproduction in any medium, provided that the use is non-commercial, that no modifications or adaptations are made, and that the original author(s) and the source are credited.

History
Received: 07 Mar 2023
Revised: 25 Aug 2023
Accepted: 24 Oct 2023
e-Available: 16 Nov 2023

Keywords
Cybersecurity standards, Validation and verification, Cybersecurity testing, Best Practices

Citation
Roberts, A., Marksteiner, S., Soyturk, M., Yaman, B. et al., "A Global Survey of Standardization and Industry Practices of Automotive Cybersecurity Validation and Verification Testing Processes and Tools," *SAE Int. J. of CAV* 7(2):199–213, 2024, doi:10.4271/12-07-02-0013.

ISSN: 2574-0741
e-ISSN: 2574-075X

Downloaded from SAE International by Tallinn University of Technology, Friday, June 13, 2025

1. Introduction

UNECE's regulation 155 (UN R155) [1] requires a structured approach to cybersecurity engineering of automotive systems, using a cybersecurity management system. This regulation is mandatory for compliance with automotive type approval within many of the most important automotive markets including Europe, Korea, and Japan. Without compliance to the UN R155, original equipment manufacturers (OEMs) may currently not commission new models, and, from mid-2024, will be restricted from selling to these markets. Therefore, OEMs are motivated to comply with this regulation due to the financial risk of losing market access due to noncompliance. Closely related to UN R155 is the ISO/SAE 21434:2021—Road Vehicles—Cybersecurity engineering, which is commonly accepted as the guiding standard for automotive cybersecurity [2]. UN R155 requires automotive manufacturers to have for their automotive product a Cybersecurity Management System (CSMS). The ISO/SAE 21434:2021 provides, so far, the only global standardized approach for development of an automotive CSMS (however, it is not explicitly mandatory that a CSMS follows that standard). UN R155 and ISO/SAE 21434:2021 require the structured measures to be verified and documented in a comprehensible and replicable manner using structured testing procedures. However, the details of how to conduct testing applicable to the requirements of UN R155 for type approval and to the standard expected for automotive risk management are mainly left to technical services, vendors, and suppliers. The global standards (including ISO/SAE 21434:2021) only recommend testing methodologies at a very high level (i.e., functional testing, vulnerability scanning, fuzz testing, penetration testing), and provide suggestions for test targets (e.g., checking for exposed debug interfaces, the presence of a secure boot mechanism, usage of encryption in communications, etc.). The complexity of vehicular systems, in conjunction with a diverse ecosystem of standards and procedures make it infeasible to define a solid, standardized testing procedure that spans over the whole (in-homogeneous) system and over the whole life cycle. The development of standardized processes is further challenged, as each large OEM has its own established procedures and guidelines, partially stemming from internal design and coding guidelines as well as from procedures from adjacent domains such as functional safety testing. There also exists a lack of literature that explores the state-of-the-art of automotive cybersecurity testing and how the global standards are being implemented regionally and how industry is developing its cybersecurity testing programs. To confront these challenges, the main idea of this research is to provide a starting point on identifying test targets and testing methods from a global and regional perspective, as well as exploring the usage and applicability of such methods currently used in the automotive industry. To this end, the contributions of this research are as follows:

- We conducted a state-of-the-art analysis of automotive validation and verification testing (V&V) for global and regional automotive cybersecurity standards and regulations.
- We conducted a survey of tools and practices commonly used by manufacturers and admission bodies and analyzed the development of cybersecurity test tools and procedures.
- We discussed the findings of the state-of-the-art and survey and analyzed the progress of the adoption of UN R155.

2. Methodology

The initial stage of the study focused on establishing the standards and regulatory environment for V&V testing of key global automotive regions. The central questions used to guide the research were:

- **RQ1** What is the state-of-the-art for automotive cybersecurity V&V standards?
- **RQ2** How have these standardization approaches been transposed to national regulation?
- **RQ3** What are the V&V testing processes, procedures, and tools used by industry?

These questions enable the extrapolation of key areas of interest for automotive cybersecurity V&V:

- Are there variances between regions in the implementation of regulation and national initiative developed to improve V&V testing, and if so, why?
- What are the key trends for V&V testing adopted in industry? What can these trends tell us about the evolving nature of V&V testing to meet technology innovation?

To answer these research questions, analysis was conducted on three data sources (see Table 1): (1) literature from government authorities, industry, and standardization groups, (2) expert knowledge derived from open-format workshops with regional representatives from a global mobility testing industry working group, and (3) an academic literature from key conferences in the automotive cybersecurity field. The purpose of the academic literature review is to provide a brief overview of the key trends as they relate to ISO/SAE 21434:2021.

2.1. Related Work

There have been numerous reviews of automotive cybersecurity standardization during and after the drafting of ISO/SAE 21434:2021 and the UNECE Regulation R155. Macher et al. [3] first review in 2019 found two predominant challenges of standardization of automotive cybersecurity testing. First, the cross-relations between standards, guidance, recommendation, and regulation created a complex environment that was difficult to interpret. Second, select automotive technologies were governed by diverse standards. An example was given of OBD-II interface, which is mentioned in hardware security and certificate standardization documents. However, the

Downloaded from SAE International by Tallinn University of Technology, Friday, June 13, 2025

© International Alliance for Mobility Testing and Standardization (IAMTS)

TABLE 1 Data sources for survey of standardization efforts for automotive cybersecurity V&V testing.

Review	Data source
Literature review of national standards and regulations	• Official government documents (legislation, govt. department documents) • Automotive and transportation reports and standardization reports • Academic literature
Industry survey	• Open format workshops with regional representatives from EU, China, Japan, and North America • Written survey with structured questions
Academic survey	• Literature from automotive security research in academia and standardization body journals

certificate standardization was legacy and was written in 2006, at which, advances in hardware security were not apparent. The second standardization review by Schmittner and Macher [4] in 2020 focused on the draft [5] of the ISO/SAE 21434:2021 standard. In addition to lauding the effort to contribute a common framework and language for automotive cybersecurity, shortcomings identified included ambiguity in descriptions of processes and approaches and the difficulty in providing a standardized context for diverse methods, guidelines, and best practices. Schober and Griessnig [6] mapped the cross-relations of automotive cybersecurity regulations (UNECE No. 155 and 156) and standards (ISO/SAE 21434:2021, ISO PAS 5112, ISO 24089). As this study was written at the initial release of ISO/SAE 21434:2021 and before UNECE R155 and 156, the national level initiatives to support innovations for automotive cybersecurity testing were not captured.

3. Global Regional Perspectives on Automotive Product V&V Testing Standardization and Regulation

3.1. Attack Automotive Product V&V Testing Standardization

As a standard released in 2021, ISO/SAE 21434:2021 [7] brings a specification and a framework for cybersecurity risk management in different phases of product lifecycle: concept, development, production, operation, maintenance, and decommissioning of electrical and electronic systems. While covering the whole engineering process of road vehicles' cybersecurity, the standard also mentions cybersecurity testing by emphasizing its importance and providing a high-level guidance. Worth noting, the document doesn't provide a detailed analysis for the testing methodologies, processes, and tools. Further, the standard brings description and distinguishes between the verification and validation. Because of the lack of test-related details in ISO/SAE 21434:2021, WG11 (Cybersecurity working group) under ISO/TC22/SC32 (Committee of Electrical and Electronic Components and General System Aspects) has proposed ISO PWI 8477, which is a new standardization project for automotive cybersecurity verification and validation. This project is intertwined with a second project: "ISO/SAE PWI 8475: Road vehicles—Cybersecurity Assurance Levels (CAL) and Target Attack Feasibility (TAF)," which is targeted to define automotive cybersecurity assurance levels (CALs) and target attack feasibility (TAF), whereby the CALs are focused on engineering assurance and the TAFs are on the expected strength of technical controls. However, there is not yet (as of June 2022) an official standards project, any results are therefore pending. The standards document SAE J3061_202112 (Cybersecurity Guidebook for Cyber-Physical Vehicle Systems) [8] contains an appendix regarding the existing security test tools. Another initiative from SAE International, which is in progress, is the J3061-2 (Security Testing Methods) [2]. The document has been issued by the Vehicle Cybersecurity Systems Engineering Committee with the aim of providing a detailed analysis on the security testing methods on both hardware and software.

A global regulation initiative on automotive cybersecurity is brought recently within an addendum to UNECE 1958 treaty (Regulations 141-160), namely UNECE R. 155 [1] and 156 [9] for automotive cybersecurity. These regulations have a direct impact on OEMs and suppliers as the compliance to UNECE's homologation regulations are fundamental for the automotive type approval process and product development for the market. UNECE Regulation No. 155 (–legally binding document ECE/TRANS/WP.29/2020/79 [10]) mandates the installment of a CSMS as defined in ISO/SAE 21434:2021 [7] to ensure an accompanying cybersecurity process to be executed during the automotive system development lifecycle. In the document, the OEM is required to verify the effectiveness of implemented cybersecurity measures by testing and the approval authority shall refuse the type approval if this cannot be demonstrated including the adequateness of the testing procedures themselves. Lastly, the authority by itself shall also verify the effectiveness of security measures by testing, especially concentrating on the high-risk samples. With the increased threads on cybersecurity of automotive systems due to increased complexity and connectivity; there are initiatives brought by the governments for regulation and standardization. It is seen that it is a general tendency by the governments to prepare the industry for the regulations, with guideline documents on how to properly implement and test the cybersecurity mechanisms (e.g., [11] and [12]). While most of the regulation initiatives across the governments regarding the V&V are still in similar phases of preparation; there are some issuances of documents regarding the type approval by the ministries of Korea and China (see subsections Republic of Korea and China under

Downloaded from SAE International by Tallinn University of Technology, Friday, June 13, 2025

Section 3.3.1). In the U.S., the government encourages the industry to collaborate with the regulation activities by commenting on the documents published by the agencies. Regarding the standardization, there are two remarkable initiatives brought by the national standardization organizations of China and Japan, which brought some practical standards on cybersecurity testing and verification (see Section 3.3.1).

3.2. Automotive Academic Survey of V&V

Current trends in academic literature focus on the following areas:

- Novel vulnerability testing of intelligent vehicular technologies and autonomous, self-driving control algorithms.
- Methods for automating cybersecurity testing.

Novel attacks on intelligent vehicular and self-driving technologies focus on the advanced hardware technologies that support perception (LiDAR, camera, radar), localization (LiDAR, GNSS), and vehicular communication [vehicle-to-vehicle (V2V), vehicle-to-infrastructure (v2x)]. Testing is predominantly conducted in high-fidelity digital twin simulation environments and progressively, real-world environments and proving grounds. Tools common in testing of these systems include adversarial neural networks that generate malicious robust physical invariants to perturb object detection and semantic segmentation, fuzzers for protocol vulnerability assessment [13, 14], and, in intelligent vehicles, to send malicious unsanitized sensor telemetry input to impact LiDAR [1, 15], radar, and inertial measurement sensors [16, 17]. White-box testing tends to be more popular for testing of neural networks due to the complexity of understanding the impact of attacks of black-box testing and to optimize testing based on knowledge of the learning model. Automation of cybersecurity testing has focused on aligning fuzz testing techniques with contemporary software development processes. Fuzzing approaches are being developed, which incorporate guidance of the ISO/SAE DIS 21434 to utilize threat and risk assessment (TARA) and cybersecurity assurance levels (CALs) to systematically identify and prioritize attack vectors [18]. Novel methods for testing are being explored on digital twin, digital replications of embedded systems, to understand attack vectors and resultant impacts in a safe, and repeatable and agile test environment [19, 20].

3.3. National Regulatory and Standardization Approaches for Automotive Product V&V Testing

Each signatory of UNECE R155 is required to transpose this regulation into national legislation. As approaches to cybersecurity testing of critical infrastructure differ it is important to understand how national governments are transposing UNECE R155 into their respective ecosystems and how they are supporting the introduction of regulations with initiatives to assist industry and authorities. It is also observed that, despite China is not a contracting party of the UNECE WP.29 1958 Agreement [21] (hence not obliged to follow UNECE R155); the national government perform similar activities referring to ISO/SAE 21434:2021. In North America, situation is different due to the performed system of self-assessment in that region. Despite this, there are national activities with respect to ISO/SAE 21434:2021.

To elucidate this, two components of national approaches to automotive cybersecurity testing are analyzed: (1) governance and implementation of regulation and ISO21434:2021 and (2) national initiatives with regard to automotive V&V testing.

3.3.1. Asia

China

Governance and Implementation of Regulation and ISO/SAE 21434:2021 The Chinese market has seen an emergence of self-driving and interconnected technologies for vehicles. Due to this, the Chinese government ministries are focused on developing policies for cybersecurity and data security of intelligent and connected vehicles (ICVs). To support these policies, corresponding standards committees are developing national standards, of which the majority still are in draft version. In particular, three ministries work in the field of cybersecurity and data security of ICV: the Ministry of Industry and Information Technology (MIIT) and Cyberspace Administration of China (also called Office of the Central Cyberspace Affairs Commission), and the Ministry of Natural Resources.

In late 2021, the MIIT has published two notices [22, 23] to address the security requirement of connected vehicles. In these notices, it mandates that both cybersecurity and data security of connected vehicle must be fully considered before going to market. Building a complete vehicular security standard system is also prescribed to all subdepartments, organizations, and companies. Meanwhile, a mandatory standard for vehicle cybersecurity and technical requirements for vehicle cybersecurity has been issued [24]. Furthermore, ISO/SAE 21434:2021 is being converted to Chinese national standards as well.

National Initiatives With Regard to Automotive Product V&V Testing For general technical security requirements, the National Information Security Standardization Technical Committee (NISSTC) released GB/T 40861-2021 [25] on October of 2021, which involved the security of software, electrical and electronic hardware, data, onboard communication, and V2X communication. Furthermore, the authenticity, confidentiality, integrity, availability, access control, anti-repudiation, auditability, and preventability should be considered to the corresponding

Downloaded from SAE International by Tallinn University of Technology, Friday, June 13, 2025

security system, if applicable. Compared to other standards, this standard provides a more complete technical requirement of in-vehicle security. Some standards released by NISSTC focus on the technical requirements as well as test methods of specified system and component. Standard GB/T 41578-2022 [26] addresses in-vehicle charging system and corresponding communication security. It further specifies detailed test methods at hardware, software, data, and communication aspects. GB/T 40856-2021 [27] concerns the security test methods for hardware, communication, operation system, application, and data. GB/T 40857-2021 [28] addresses hardware, software, communication, and data security for CAN gateway, ethernet gateway, and hybrid gateway. GB/T 40855-2021 [29] involves on-board terminals security, communication security, and platform security in the scope. With regard to different kinds of security, standards also provide a few general best practices for testing. For hardware, this includes checking for exposed debug interfaces and their authentication mechanisms, the disclosure of the PCB wiring and design, and for backdoors. For software, checks for secure boot and software integrity, access control, logging mechanisms, as well as vulnerability scans are recommended. The data should be checked for susceptibility to tampering, confidentiality on export, collecting after user approval, sensitive information protection, effectiveness of its deletion, as well as its security during transmission. Communication links should prove their authentication, integrity confidentiality availability, and non-repudiation.

Japan

Governance and Implementation of Regulation and ISO/SAE 21434:2021 Japanese METI (Ministry and Economy, Trade and Industry) published a document about cybersecurity measures for autonomous vehicles in 2018. This document describes the schedule for implementing ISO/SAE 21434:2021. First, JASPAR (Japan Automotive Software Platform and Architecture) collaborates with other countries to establish the standard while suggesting rules and policies that fit in Japanese automotive environment. While developing ISO/SAE 21434:2021, METI and MLIT (Ministry of Land, Infrastructure, Transport and Tourism) create guidelines that describe requirements to develop and operate automotive vehicles, with some governmental organizations such as JASPAR. Besides, METI creates a more concrete guideline for testing and validation/certification of autonomous vehicles collaborating with organizations in industrial sector such as IPA (Information Processing Agency). Until now, MLIT has published guidelines for requirements of autonomous vehicle development like [30] (Japanese). Also, IPA has published and revised more practical guidelines such as [31]. This guideline includes threat analysis and possible measures in a development cycle, namely management, planning, development, and operation. National standards are determined by organizations such as JAPSAR, based on the international standards. The national standards describe requirements that the industry must meet in the development process against assumed security threats. Especially, they have formulated evaluation guide for ECU and hardware/software vulnerabilities. JASO TP-15002 guideline is an evaluation guideline for automotive information security analysis. Japan Automotive Software Platform and Architecture (JASPAR) is a collaboration project of engineers from the automotive industry. The aim of JASPAR is [32]: "identify common issues that will be faced in the future in the car electronics sector, and then undertake standardization initiatives aimed at resolving those issues, creating common objectives across the entire automotive industry."

National Initiatives With Regard to Automotive Product V&V Testing The JASPAR project provides reference architectures for secure design of automotive components and verification testing. The standards are focused on areas of cybersecurity of car electronics where there are gaps in other available standards and areas that are a priority for the Japanese automotive industry. These include software-over-the-air updates, ECUs, CAN-FD, secure communication, and vehicular messaging. JASPAR project details a list of standards applicable to cybersecurity testing of automotive products: TD-CST-4—ECU Penetration Testing Guide Version 1.0, ST-CST-1—ECU Vulnerability Test Requirements Ver.1.1, STOTA-09—OTA Software Update Compliance Test Specification OTA Master Ver.1.0, ST-OTA-10—OTA Software Update Compliance Test Specification—Target ECU Ver.1.0 [32].

Republic of Korea

Governance and Implementation of Regulation and ISO/SAE 21434:2021 There are two main actors in Korea for type approval and certification of vehicles Ministry of Land and Infrastructure, Transport (MOLIT) and Korea Automobile Testing & Research Institute (KATRI) [33]. There are two regulations that pertain to the testing and evaluation of automotive:

- Korea Motor Vehicle Safety Standard (KMVSS)—Technical Regulation
- Korea Vehicle Management Act (Self-Certification system and Safety Standards for Motor Vehicles)

In June 2020, MOLIT established the UNECE R155 international standards for automotive cybersecurity as the main content for recommendations for ROK automotive manufactures. The central component being that the automotive manufacturer has a cybersecurity management system (CSMS) and demonstrate that automotive cybersecurity is managed accordingly. To integrate UNECE R155 local laws and regulations will be amended as appropriate [33]. MOLIT plans to issue the Automotive Cybersecurity law and safety/security regulation in 2022. Until that time, they will have published recommendations and guidelines to fill the gap between the practice of automotive company and the requirements imposed by the registration such as Korea Motor Vehicle Safety Standard (KMVSS) and Korea Vehicle Management Act. The approach taken by MOLIT is to ease

Downloaded from SAE International by Tallinn University of Technology, Friday, June 13, 2025

the new policy implementation and adoption recommendations step-by-step. Currently, the differences of UN R155 and the ROK implementation are that ROK extends the R155 to their self-certification approval in addition to type approval, manufacturers obligation to report are focused on data sharing between manufacturers, and administrative matters (procedures, document, penalties) and matters relating to type approval (CSMS certification, DETA data sharing) are yet to be included in the implementation [33]. As one of the recommendations, MOLIT announced guidelines for security of autonomous vehicles on December 15, 2020. The guidelines include (1) Ethical Guidelines for Self-Driving Vehicles, (2) Automobile Cybersecurity Guidelines, and (3) Level 4 Autonomous Vehicle Manufacturing/Safety Guidelines, which provide basic directions for ethics and safety [33]. Among that, Automobile Cybersecurity guidelines introduced recommendations for security policy directions so that automobile manufacturers can develop a cybersecurity system in preparation for the implementation of the security standards to be issued in 2022 [34]. The recommendations proposed in the guidelines are the following:

- Security management such as a process for identifying, evaluating, classifying, and managing security threats must be established within the manufacturer's organization and share relevant information.
- Vehicle security threat identification, evaluation, security measures, and sufficient security-related pre-tests must be performed. Note that security measures include cyberattack detection and prevention measures, risk monitoring support measures, data forensics support measures for cyberattack analysis, and the like.

To support the implementation of R155 as part of domestic regulations, MOLIT has planned to implement an Automotive Cybersecurity Support and Response System. This system consists of an automotive cybersecurity committee to coordinate initiatives including the foundation of an automotive security center. The role of the Automotive Cybersecurity Support and Response System is to provide cybersecurity test and evaluation and enforcement support, support the private sector with the development of automotive technologies, provide cybersecurity incident response, and support for the automotive sector [33].

3.3.2. North America

United States of America

Governance and Implementation of Regulation and ISO/SAE 21434:2021 In the U.S., National Highway Traffic Safety Administration (NHTSA) is the responsible entity under the U.S. Department of Transportation (U.S.DOT), which issues Federal Motor Vehicle Safety Standards (FMVSS) to regulate and standardize the requirements for the safety of motor vehicles [35]. The agency undertakes the responsibility of standardization and regulation of automotive cybersecurity in the U.S. while conducting research in order to address the challenges in the area [36]. To provide a comprehensive and systematic standardization and regulation process, the agency involves the industry in the regulation and standardization process by encouraging the formation [37] of Auto-ISAC [38] and receiving comments on the publications/reports that are published by the agency [39]. Currently, there are no standards or regulations for automotive cybersecurity testing and verification, which is brought by the NHTSA. However, in 2016, the agency published a non-binding document describing guidelines and best practices for automotive cybersecurity [40], which is revised in 2020 concerning the ISO/SAE DIS 21434 draft standard and a draft version has been published (2020 draft) [11]. According to the comments brought on the draft, a pre-final version has been released in 2022 [41]. The document refers ISO/SAE 21434:2021 and NIST's Cybersecurity Framework for standardizing the cybersecurity development, maintaining, and testing process.

National Initiatives With Regard to Automotive Product V&V Testing NHTSA conducts multifaceted research on vehicle cybersecurity that leverages NIST's cybersecurity framework [42] and aims to collaborate with the industry to address the challenges in vehicle cybersecurity. NHTSA's best practices documents include recommendations for automotive cybersecurity testing and documentation. Those practices defined in [41] are as follows:

- Cybersecurity testing, including penetration testing should be implemented as a part of the development process.
- Qualified testers who have not been a part of the development process should be included in the testing phases.
- Identified vulnerabilities during cybersecurity testing should be analyzed; the vulnerability and how the vulnerability is managed should be documented.
- All commercial-off-the-shelf and open-source software components used in vehicle ECUs should be evaluated by the manufacturers in order to identify the vulnerabilities.

For addressing the need for effective information sharing across the industry, NHTSA encouraged the formation of the Auto ISAC, a community established by partners from the various domains of the industry. In collaboration with the Alliance of Automobile Manufacturers (Auto Alliance) and the Association of Global Automakers (Global Automakers), the community published a set of best practices documents on automotive cybersecurity [43]. One of these documents, "Security Development Lifecycle," covers the security needs for the development process and distributes the testing process into the phases of development as follows [44]:

i. **Design**: This phase is where a high-level test plan can be constructed, which identifies:
    - The best security verification methods (e.g., design review, manual code review, automated code analysis, component/unit testing, bench and vehicle penetration testing).

Downloaded from SAE International by Tallinn University of Technology, Friday, June 13, 2025

- Needed testing tools including special build components and infrastructure support.
- An evidence sheet with details of software, hardware level, date, pass/fail status, notes on failures or unexpected behavior person running the test and approver, and others as necessary.

ii. **Implementation**: Secure implementation requires testing and verification in both hardware and software levels. The methods for ensuring at the hardware level:

- Confirmation reviews or assessments
- Penetration tests

At the software level:

- Code reviews
- Automated code analysis
- Penetration testing

iii. **Testing and Validation:** This part defines the whole process of testing through phases of the development lifecycle:

1. Cybersecurity Testing: The actual testing process is done during the implementation and post-implementation phase, which evaluates the proper working of safeguard mechanisms and identify potential vulnerabilities that leads to residual risk assessments.
2. Internal Cybersecurity Sign-off Process: The sign-off process includes the testing process, which verifies the system is secure enough to withstand the previously assessed threats. This process should include the overall test plan, performed functional tests, penetration tests, source code audits, and so forth.
3. Residual Risk Assessments: Residual risk assessments can be done as a part of the development lifecycle on a periodic basis as the known residual risks evolve over time by the discovery of new attack methods or cost reduction due to newer/cheaper tools.

Canada

Governance and Implementation of Regulation and ISO/SAE 21434:2021 In Transport Canada's Vehicle Cybersecurity Strategy, the Canadian Department of Transport is responsible for monitoring the work of the National Research Council Canada's Automotive and Surface Transportation Centre. The Automotive and Surface Transportation Centre engages in research and testing related to advanced vehicle technologies. Examples include examination of cybersecurity vulnerabilities in connected features, mapping, and connectivity for automated driving. The testing and evaluation of cybersecurity is closely tied to applicable motor vehicle safety and data privacy legislation [45].

National Initiatives With Regard to Automotive Product V&V Testing The Canada Vehicle Cybersecurity Guidance [45] provides technology-neutral and non-prescriptive guiding principles for the incorporation of cybersecurity throughout the vehicle lifecycle. The guidance promotes the importance of international standards such as ISO/SAE 21434:2021 and other related functional safety standards. The guide provides a descriptive overview of the context of cyber-attacks to vehicular systems and in particular that more advanced attacks tend to be associated with "white-hat" cybersecurity research, while real-world, cyber-criminal threat actors make use of the data-driven ecosystem of vehicular technologies to comprise attacks on back-end systems and systems that generate and store telemetry. To this end, the guide recommends the implementation of layered security controls (known as defense-in-depth), privacy protection, and information protection procedures and testing of data security, secure external vehicle communications, identity management and access control, secure software development, secure updates, and the extended vehicle environment. Cybersecurity testing is recommended to be conducted throughout the vehicle lifecycle. Penetration testing is mentioned as an essential part of security auditing. Cybersecurity testing and validation methods are not explicit in the guidance provided by Transport Canada. Transport Canada provides tier 1 and 2 automotive suppliers with a self-assessment tool: the Vehicle Cybersecurity Assessment Tool (VCAT). The VCAT is a self-assessment questionnaire applicable for all vehicle types with varying levels of connectivity and automated features. The self-assessment questionnaire assists with evaluating the cybersecurity performance and resilience of vehicles and vehicular components. The VCAT will provide a score, measuring cybersecurity posture, as well as recommendations for mitigations [45].

3.3.3. Europe

The European Union has a diverse range of regulatory initiatives for cybersecurity of the digital marketplace, which impact upon automotive product development. The EU Cybersecurity Act (CSA) is the predominant form of regulation for cybersecurity in the EU market. Among the range of important initiatives, the CSA establishes a framework for certification of ICT products for cybersecurity called the Common Criteria-based European Candidate Cybersecurity Certification scheme (EUCC). The aim of the scheme is to enable, for the consumer, transparency and awareness of the level of assurance for cybersecurity of a digital product. The EUCC is still in development and its impact on the automotive sector is yet to be detailed [46].

The EU Cyber Resilience Act (CRA) [47] is currently being developed. This regulation will focus on providing common cybersecurity rules for manufacturers and vendors of tangible and intangible digital products and ancillary services. The CRA regulation envisages a process for the digital product cybersecurity assurance where essential baseline security requirements are defined, which can be applied selectively according to a risk management assessment of a device's intended use, considering the ecosystem or

Downloaded from SAE International by Tallinn University of Technology, Friday, June 13, 2025

"operational environment" in which the device will be placed. The products to be governed by the CRA include [47]:

- Connected product: A finished product that is intended to communicate directly or indirectly over the internet.
- Finished product: A product usable for its intended functions without being embedded or integrated into any other product. Components of a device, such as a processor or a sensor, should be outside the scope as security functionalities need to be assessed holistically.

In the public submissions to the CRA regulation, automotive industry bodies (European Automobile Manufacturers' Association, European Association of Automotive Suppliers, TÜV Association) pointed to other existing legislation as impacting automotive cybersecurity [47]:

- Type-approval: UN R155 and 156
- Radio Equipment Directive (2014/53/EU) and its delegated act (2022/30) (For Connected Vehicles)
- NIS 2 Directive (2020/0359(COD))

As the EU CSA is in policy implementation phase and the EU CRA is in policy conception phase, there is a sparsity of detail as to how automotive technologies will be validated and verified for cybersecurity.

Germany

Governance and Implementation of Regulation and ISO/SAE 21434:2021 In Germany, the Federal Motor Transport Authority (Kraftfahrt-Bundesamt—KBA) is responsible for bringing UNECE R155 into national legislation by issuing guidance and legally binding rules for application and review of the regulation [48]. This application document specifies testing verification procedures by document review, as well as functional security and penetration testing of a technical service (e.g., TÜV) under witness/supervision of a neutral party (KBA or an authorized body).

National Initiatives With Regard to Automotive Product V&V Testing As the EU CSA is in policy implementation phase and the EU CRA is in policy conception phase, there is a sparsity of detail as to how automotive technologies will be validated and verified for cybersecurity.

Germany Governance and Implementation of Regulation and ISO/SAE 21434:2021 In Germany, the Federal Motor Transport Authority (Kraftfahrt–Bundesamt—KBA) is responsible for bringing UNECE R155 into national legislation by issuing guidance and legally binding rules for application and review of the regulation [48]. This application document specifies testing verification procedures by document review, as well as functional security and penetration testing of a technical service (e.g., TÜV) under witness/supervision of a neutral party (KBA or an authorized body).

National Initiatives With Regard to Automotive Product V&V Testing The Quality Management Center (QMC) of the German Association of the Automotive Industry (Verband der Automobilindustrie—VDA) issued a supplement to the process management specification Automotive SPICE (Software Process Improvement and Capability Determination), which conforms with ISO 15504 [7]. This supplement, called Automotive SPICE for Cybersecurity Engineering [49], defined a set of process steps dedicated to cybersecurity engineering that is to be used in conjunction with the current Automotive SPICE process; namely:

- SEC.1 Cybersecurity Requirements Elicitation
- SEC.2 Cybersecurity Implementation
- SEC.3 Risk Treatment Verification
- SEC.4 Risk Treatment Validation, and a new management step
- MAN.7 Cybersecurity Risk Management, as well as expanding the acquisition step
- ACQ.2 Supplier Request and Selection—In particular, the risk treatment verification prescribes a specification that is suitable to provide evidence for compliance with the security requirements and the design implementation and component integration is to be tested using defined test cases (according to a verification strategy that is derived from the requirements and implementation). The corresponding best practices provides hints on what to test:
    - Requirements-based testing and interface testing on system and software level,
    - Check for any unspecified functionalities,
    - Resource consumption evaluation,
    - Control flow and data flow verification, and
    - Static analysis; for software: static code analysis, e.g., industry-recognized security-focused coding standards. As well as some testing techniques (non-exhaustive)
    - Network tests simulating attacks (non-authorized commands, signals with wrong hash key, flooding the connection with messages, etc.), and
    - Simulating brute force attacks,
    - Audits,
    - Inspections,
    - Peer reviews,
    - Walkthroughs,
    - Code reviews.

Test cases could be derived by:

- Requirements analysis,
- Building equivalence classes,
- Testing edge cases (boundary values),

Downloaded from SAE International by Tallinn University of Technology, Friday, June 13, 2025

- Experience-based testing. The specification also proposes to establish bidirectional traceability between the verification activities and the system design. Analogously, the risk treatment has to be validated, which means the adequacy of the implemented measures (whereas the verification assures the compliance of the measures with the requirements). The validation includes activities to also detect priorly unidentified vulnerabilities (e.g., through penetration testing), while the methodology is similar to the verification.

France

Governance and Implementation of Regulation and ISO/SAE 21434:2021 In 2021, the French legislature incorporated UNECE R155 & 156. The regulatory environment in France is conducive of close cooperation with the EU. The Ministère de la Transition écologique et solidaire is the supervising authority responsible for vehicle type approval. Association Française de Normalisation (AFNOR) is responsible for automotive standardization, including cybersecurity standards. The Agence nationale de la sécurité des systèmes d'information (ANSSI) is the primary agency responsible for cyber expertise and its role involves monitoring the cyber threat landscape, raising awareness of the necessary protections required in the digital environment of France through best practices and standardization and providing technical advice and assistance including cyber incident response through CERT France (CERT-FR) [50]. Among numerous measures contained in the Critical Information Infrastructure Law 2013, ANSSI can impose technical and organizational requirements for security and trigger audits. Recent domestic legislative updates in France reflect the widespread adoption in the EU of the EU Cybersecurity Act and other related measures [51].

National Initiatives With Regard to Automotive Product V&V Testing The French Ministry of the Interior (Ministère de l'Intérieur) issued a position paper on automated driving (L'automatisation des véhicules) [52] that contains an annex covering cybersecurity (Annexe 9: la Cybersécurité). Regarding testing, this annex contains the notion to use risk analyses, compliance audits, and penetration tests. The ANSSI states in an analysis of contributions for a—generic, but also including vehicles—cybersecurity certification scheme for the usage of static source code analysis tools, vulnerability scanners, automation of configuration audit, and protocol fuzzers for verification [53], which is, however, a very high-level recommendation.

United Kingdom

Governance and Implementation of Regulation and ISO/SAE 21434:2021 Department of Transportation (DfT) and British Standardization Organization (BSI) are the main entities in the United Kingdom toward the regulation and standardization of the automotive cybersecurity, including the cybersecurity for connected and autonomous vehicles (CAVs). DfT accommodates a center called "Centre for Connected and Autonomous Vehicles," which serves also as a part of Department for Business, Energy & Industrial Strategy. The center conducts research and publishes documents regarding the safety and security of CAVs. The Centre for the Protection of National Infrastructure (CPNI) is another entity that contributed on the research for security of CAVs [54]. In 2017, DfT, CPNI, and Centre for Connected and Autonomous Vehicles published a guidance document [54], which explained the cybersecurity needs of automotive industry in eight principles. In 2021, BSI published a whitepaper [55], which defines the cybersecurity threat vectors for connected vehicles and how to meet the compliance requirements defined by the ISO/SAE 21434:2021. The paper includes an overview of ISO/SAE 21434:2021 and BSI's E2E automotive cybersecurity model, which is compliant to a set of international standards including ISO/SAE 21434:2021.

BSI PAS 1885:2018 [12] is a standards document that details the fundamental principles of cybersecurity across the vehicle's lifetime. The document provides principles that focus on organizational management of cybersecurity risks, management of the supply chain, third parties and subcontractors, and recommendations for cybersecurity design, resilience, and response measures. Principle 6, "The security of all software is managed throughout its lifecycle," prescribes a list of recommendations for testing and evaluation of vehicular software. In summary, the recommendations are:

- Open source or third-party software should be reviewed for vulnerabilities using formal code inspection reviews. Automated tools should be used to analyze the structure and security of the code.
- Configuration and management control should include evidence of testing, including test scenarios and results. Also, unresolved test defects, deficiencies, and anomalies should be documented.
- Updates shall be tested.

There is also an effort put by the British government toward the adaptation of CAVs. In 2019, the Centre for Connected and Autonomous Vehicles has started a program, called CAVPASS, in order to implement standardization, testing, and monitoring processes to ensure the resilience of CAVs against cyberattacks [56]. Zenzic is another organization founded by the government and industry in order to embrace the cybersecurity and safety challenges brought by the Connected and Autonomous Mobility (CAM). The organization published a feasibility report in 2020 [57], which stated the outcomes of several projects. The report included a part regarding the measurement and monitoring the cyber resilience, mentioning the digital twin technology for validation, assurance, and certification of CAVs.

4. Processes and Tools Used in The Industry

In order to examine which processes and tools are used in the industry, we issued questionnaires to experts in the field,

Downloaded from SAE International by Tallinn University of Technology, Friday, June 13, 2025

consisting of members of OEMs, suppliers, and automotive engineering companies. The questions targeted in collecting common practices on what is to be tested (test targets), how to test (standards usage, test types, and test derivation), and how to support the testing (test tools).

4.1. Test Targets

E/E components remain the predominant areas of focus for SUT due their importance for functionality of the vehicle. Due to the preponderance of connected vehicular technologies, communication protocols are an area of concentric concern for cybersecurity testing. Emerging SUTs include the end-to-end driving technology which supports autonomous-assisted and autonomous driving. Third-party service providers for verification and validation are popularly used due to their existing experience of testing and certification, alignment with ISO/SAE 21434:2021 and other standards which emphasize the use of third parties for independent verification and validation, and lack of available skills for cybersecurity testing of automotive products. A majority of respondents answered that they have an established interface agreement for cybersecurity testing. Most OEMs follow a document-based audition process in their verification and validation agreement.

4.2. Standards Utilization

Overwhelmingly, ISO/SAE 21434:2021 is used for cybersecurity verification and validation. Respondents also mentioned well-established, complimentary standards such as ISO/IEC 15408 (Common Criteria) and ISO/IEC 27034 (Application Security Standards). The testing process for SUTs are mainly conducted on a case-by-case basis. The limited use of test matrix and standard test sets can be seen as due to a variety of reasons including repeatable test processes cannot be ubiquitously applied to diverse range of automotive technologies, level of integration, and architecture requires testing to be approached on a case-by-case basis, lack of development, and adoption of testing metrics and criteria, cybersecurity testing is still developing and there is a lack of adoption of testing processes that support automation and repeatable testing. OEMs conduct functional testing, vulnerability scanning, penetration testing, and fuzz testing. All of these test procedures are recommendations of ISO/SAE 21434:2021 and are essential as part of an automotive cybersecurity testing program. Specifications coverage is the most popular method to measure and maximize test coverage of the SUT. This aligns with product development lifecycle and the focus on assurance for the intended functionality of the automotive component. Emerging methods include considerations for the requirements from UN R155.

4.3. Types of Testing

Our survey results show that our respondents practice various types of testing during different stages of their development lifecycle. These are (1) fuzzing, (2) penetration testing, and (3) functional testing. This section compiles these methods by describing and referring to the phases of development that each type of testing utilized. We also give further detail by adding other methods that are applicable for automotive cybersecurity testing, which are found in the literature. These are (3) model-based security testing, (4) risk-based security testing, and (6) vulnerability scanning.

Fuzzing: Fuzzing, or fuzz testing, refers to subjecting the software system (or components individually) to a large volume of invalid, unexpected, or random inputs that are known as "fuzz." By exposing the executable software to a wide range of invalid data, vulnerabilities can be identified that are not known previously. To generate a variety of inputs that can lead the program to failure, which is a difficult process to cover all cases, there are several techniques used. One of them is to generate the input data based on the analysis of a program's coverage, behavior, and source code, another is to implement mutation techniques on the generated data according to the program's feedback from the previously fed data, or to randomly generate [58]. Fuzzing is conducted during the development and testing phases of ECUs and infotainment systems to discover vulnerabilities, software bugs, or unexpected behavior that may lead to failures.

Penetration Testing: Penetration testing is conducted to assess the security of the hardware, software, and communication systems, by mimicking real-world security attacks on the subject. It involves actively scanning and exploiting vulnerabilities in the system with methods such as injection and tampering to determine its susceptibility to unauthorized access, data breaches, or malicious activities. Penetration testing is performed during the entire development lifecycle and before deployment to identify security flaws and mitigate them before they can be exploited by attackers.

SAE J3061 and o ISO/SAE 21434 state the necessity of penetration testing and it is included as part of the best practices document published by Auto ISAC [43]. Also, a recent study [59] shows its wide usage among security testing types. It is also seen that, among different knowledge levels, black-box testing is the most preferred one for penetration testing.

Functional Security Testing: Focuses on evaluating the security features and mechanisms of the system to ensure they function as intended. It involves subjecting security properties, such as authentication, authorization, encryption, and secure communication mechanisms to test and verify their compliance with the security requirements and validate the behavior. This type of testing is applicable by both software-in-the-loop and hardware-in-the-loop testbeds, which may be utilized throughout the development [60]. Functional security testing can be conducted throughout the development lifecycle and pre-deployment stage to verify and validate the security features.

Model-based Security Testing: Model-based security testing involves creating formal or semi-formal models of a feature, and using these models to perform security analysis and verification of conformity to requirements. Models can be security properties (i.e., confidentiality, integrity, authentication, etc.), vulnerabilities, and security safeguards that are

Downloaded from SAE International by Tallinn University of Technology, Friday, June 13, 2025

being designed for the overall system, and also threats and attacks to the system [61]. This type of testing helps identify potential security weaknesses in the design and earlier phases of the development lifecycle and enables engineers to mitigate the vulnerabilities by designing robust features.

Risk-based Security Testing: Focuses on assessing the security of a system based on the potential security risks and their impact. This type of testing is based on threat analysis and risk assessment (TARA) techniques to prioritize the most critical assets, threats, and vulnerabilities for allocating testing resources accordingly [59]. Risk-based security testing considers the likelihood of an attack, its potential impact on the system, and the value of the assets at risk. It is performed throughout the development lifecycle to ensure most critical security risks are addressed.

Vulnerability Scanning: This is a systematic process to test the system for known vulnerabilities that can be exploited by known threats. This type of testing can target the source code by conducting either static or dynamic analysis to understand whether the software poses vulnerabilities due to memory usage or the interfaces for discovering unprotected entries (i.e. port scanning) [60]. Automated tools and scripts are used in this approach so that they can be implemented as part of the DevOps cycle to conduct regular and repeatable tests with each increment, during development, and after deployment (i.e., for updated software).

Two-thirds of respondents confirmed that they utilize functional testing and penetration testing within their verification and validation processes, which support the entire automotive development lifecycle. Validation activities were conducted close to the end of the product development phase and before release for post-development and consisted of analysis and testing. Verification activities were conducted during the concept and product development phase and consisted of review, analysis, and multiple rounds of penetration testing. One-third of respondents have not yet adopted the cybersecurity verification and validation processes of the ISO/SAE 21434:2021 standard.

4.4. Test Derivation

There is a couple of ways to derive test cases from a performed asset/security analysis: based on derived requirements from a model (e.g., a TARA, cf. previous section) that could also be subject to model checking; based on specifications (both standards and vendor specifications), based on the structure (i.e., the architecture—e.g., tests that verify the correctness of a security gateway's functioning), based on the experience of the respective penetration tester (i.e., trusting the right test cases to be designed to expert knowledge), or based on known faults. The respondents roughly evenly perform requirements, specification, and experience-based test derivation, while structure-based tests are significantly less (one-third) used, information is UNECE's Regulation 155 (see above in the respective section) [1]. In its Annex 5 it defines a catalogue of countermeasures that can serve as requirements that might be verified by testing. Regarding the testing methods, it is equally proliferated to use white box (full access to information about the SUT), black box (just the SUT "as is," with no additional information), and gray box (some information, mainly handbooks, API documentation, etc.) approaches. Only a minority (one-third) of the respondents claimed that they use a baseline for testing. This means a minimum set of tests generically issued to all of their SUTs, regardless of their nature. The relative majority of those uses testing the requirement specification followed by using prepared test plans, test cases, and test data and, lastly, testing the design specification and predefined generic tests for the source code itself. One specific test set mentioned is testing all wireless and wired interfaces (e.g., OBD) for their susceptibility to act as an entry vector into the vehicle.

4.5. Test Tool Categories

Respondents use a diverse range of commercial-off-the-shelf (COTS), open-source (OS), customized, and in-house (internally developed) tools in their penetration testing activities. The results show a bias toward COTS and OS tools. The respondents also identified a number of tools that were used to test recent high-profile vulnerabilities such as Blueborne (a well-known Bluetooth attack) and ROCA (cryptographic weakness). With the emphasis ISO/SAE 21434:2021 places on TARA, it is apparent that automotive cybersecurity testers are agile in developing and utilizing toolsets to keep pace with the dynamic threat environment. Table 2 categorizes specifically mentioned tools. When asked for specific tools during the phases of an attack test—pre-attack (scanning, CAN analysis, etc.), attack (exploit frameworks, etc.), and post-attack (reporting, life cycle management)—respondents answered with a variety of tools.

Table 3 provides an overview of some commonly used tools, displaying the phase that are used in reconnaissance, attack, or life cycle governance; the tool category (cf. Table 2); and the area of testing (IP/web, wireless, and in-vehicle networks as well as reverse engineering). In that context, IP Network/web testing tools refer to tools originally used in traditional IT testing, targeting network, and web-based interfaces. Currently, they are ordinarily used mainly to perform tests in automotive ethernet or on targets that have interfaces similar to traditional IT systems, e.g., infotainment head units running on an Android operating system. Wireless Automotive refers to tools to assess implementations of wireless protocol stacks that are popular in the automotive industry, most prominently Bluetooth and WiFi. In-vehicle network (IVN) tools mainly refer to tools for testing CAN bus and Automotive Ethernet environments. Lastly, reverse engineering tools are used to scrutinize binaries of automotive control systems and search for potential weaknesses inside the code by following control flows. The other axis of the table shows whether the tool is considered to be more in reconnaissance (information gathering) or attack (actual intrusion) phase of cracking a system, as well as life cycle management tools that support the security governance and help in planning tests throughout a system's life cycle.

Downloaded from SAE International by Tallinn University of Technology, Friday, June 13, 2025

TABLE 2 Tool categories.

Tool category	Description	Automotive test usage
Vulnerability assessment	Enables performance of a scan of a device or information system to discover vulnerability of the target system to known vulnerabilities.	Nmap and Nessus could be used to find open communication ports on an infotainment head unit and its vulnerabilities.
Web application	Enables analysis of the codebase of web applications and mobile device applications.	Predominantly used in the testing of infotainment systems and customer applications.
Reverse engineering	Used for analyzing the binary code of the software to identify vulnerabilities (due to memory usage, logic, etc.). Tools such as IDA and Volatility (see Table 3) are used for data extraction for analysis.	
Protocol analysis	Enables analysis of protocols to understand the architecture and identify vulnerabilities.	Used for internal (CAN, LIN, MOST, FlexRay) and external (Wireless, Radio, Bluetooth) networks.
Fuzzing	Used to assess the security of a system to unsanitized data input. This can be either randomized or targeted unsanitized data input. It is popularly used in software engineering to identify bugs in the codebase.	Fuzzing is used ubiquitously from the embedded hardware ECUs to the infotainment system, mostly through customized or in-house tools aligned with the OEM software development processes.

© International Alliance for Mobility Testing and Standardization (IAMTS)

5. Discussion

The most influential document is arguably UNECE R155 for its normative and legally binding character. This document contains a list of requirements (in an annex) that could serve as test targets. Further details are specified on a national level, as is the details of the mandated CSMS. The specification of such a system is found in ISO 21434:2021. Both documents, however, specify testing requirements at a very high level. Therefore, the ISO maintains ongoing efforts to specify test classifications, as well as V&V procedures in more detail, giving guidance for testing. As the UNECE must be adopted into national regulations, the concrete embodiments differ. Nonetheless, it is common that the level of detail is coarse, leaving much room for interpretation open for implementers. Regional standards are likewise high-level descriptive in general, focusing on engineering process topics. An exception are some standards specifically from the Asian area that give fine-grained descriptions for test procedures for single components. The research of regional standards showed no clear bias in testing procedures by region, although the underspecification leaves room for interpretation differences by both different regional authorities and implementers. What is missing globally is test implementation details for systems at vehicle level. The reason, drawn out of expert interviews, is the early stage maturity of the topic. First, details for many of the components have to emerge, before they can be tied to high-level test procedures at vehicle level. To perform testing and analysis, most

TABLE 3 Testing tools per attack phase, type, and category.

Phase	Tool	IP network/web	Wireless automotive	IVN	Reverse engineering	Tool category
Reconnaissance	Nessus	✓				Vulnerability assessment
	Nmap	✓				Vulnerability assessment
	Dirbuster	✓				Fuzzing
	Bluescanner		✓			Vulnerability assessment
	Wireshark	✓				Protocol analysis
	GNU Radio Companion		✓			Protocol analysis
	Universal Radio Hacker		✓			Protocol analysis
	CANoe			✓		Protocol analysis
Attack tools	Ghidra				✓	Reverse engineering
	Android Studio				✓	Reverse engineering
	Aircrack Suite		✓			Vulnerability assessment
	URH		✓			Reverse engineering
	Volatility				✓	Reverse engineering
	Genymotion				✓	Protocol analysis
	IDA				✓	Reverse engineering
	Burpsuite	✓				Web application
	American Fuzzy Lop				✓	Fuzzing
LCM	PTC Integrity					

© International Alliance for Mobility Testing and Standardization (IAMTS)

Downloaded from SAE International by Tallinn University of Technology, Friday, June 13, 2025

players currently use general-purpose tools that are proliferated in IT security testing (e.g., fuzzers, reverse engineering, and protocol analysis tools), as well as specialized hard- and software for automotive systems (particularly CAN buses). These tools are used primarily in a manual testing process. There is few automatic test generation and execution methodology for automotive security (such as [62]). Apart from systems for supporting the testing process by automating tasks (e.g., vulnerability scanning) and embedding this in an automated toolchain, one trend tends to be going toward model-based testing.

6. Conclusion

This research provided an overview of international and regional standards and found that the current state-of-the-art lacks proscribed detail of V&V procedures that would enable alignment within different regions and industry. The developmental nature of V&V testing was further highlighted by the industry working group responses, which demonstrated that traditional enterprise information technology and processes were used. We see, however, that there is considerable development in this area with industry identifying connected and autonomous vehicle technologies as increasing in priority for testing and the focus on developing toolsets for automotive cybersecurity testing. Furthermore, we also see a concentration of effort by national authorities to enshrine UN R.155 into the national regulatory frameworks for vehicle regulation and advocate for best practice guidelines such as those in ISO/SAE 21434.

As the UNECE regulation and its accompanying traits are fairly new (first effective only in mid-2022), there is a significant lack of experience on necessary test procedures. Practical advice will emerge in greater detail when it could be clarified how the legislation is actually handled. The same applies for standards, as pivotal initiatives (e.g., from ISO) are still in a very early project phase—with forthcoming of these endeavors more detailed specifications can be given. Dedicated, automated toolchains will follow that trail, so far incipient stages are given.

Acknowledgements

This work has been supported by the European Commission through the H2020 teaming project Finest Twins (grant No. 856602) and European Union's Horizon 2020 Research and Innovation Programme, under grant agreement No 883321 (CitySCAPE). The authors want to thank the International Alliance for Mobility Testing and Standardization (IAMTS) for their support while writing this article.

Contact Information

Stefan Marksteiner
Corresponding author
stefan.marksteiner@avl.com

References

1. United Nations Economic and Social Council—Economic Commission for Europe, "Cyber Security and Cyber Security Management System," Regulation 155, Brussels, 2021.
2. SAE International, "J3061-2 (WIP) Security Testing Methods," accessed June 28, 2023, https://www.sae.org/standards/content/j3061-2/; International Organization for Standardization and Society of Automotive Engineers, "Road Vehicles—Cybersecurity Engineering," ISO/SAE Standard 21434:2021, 2021.
3. Schmittner, C. and Macher, G., "Automotive Cybersecurity Standards—Relation and Overview," in *Computer Safety, Reliability, and Security: SAFECOMP 2019 Workshops, ASSURE, DECSoS, SASSUR, STRIVE, and WAISE, Turku, Finland, September 10, 2019, Proceedings*, Romanovsky, A., Troubitsyna, E., Gashi, I., Schoitsch, E. et al. (Eds.) (Berlin, Heidelberg: Springer-Verlag, 2019), 153-165, https://doi.org/10.1007/978-3-030-26250-1_12.
4. Macher, G., Schmittner, C., Veledar, O., and Brenner, E., "ISO/SAE DIS 21434 Automotive Cybersecurity Standard—In a Nutshell," in *Computer Safety, Reliability, and Security. SAFECOMP 2020 Workshops*, Casimiro, A., Ortmeier, F., Schoitsch, E., Bitsch, F. et al. (Eds.) (Cham: Springer International Publishing, 2020), 123-135.
5. International Organization for Standardization and Society of Automotive Engineers, "Road Vehicles—Cybersecurity Engineering," ISO/SAE Draft International Standard DIS 21434, 2021.
6. Schober, T. and Griessnig, G., "Cybersecurity Regulations and Standards in the Automotive Domain," in *Systems, Software and Services Process Improvement (Communications in Computer and Information Science)*, Yilmaz, M., Clarke, P., Messnarz, R., and Wöran, B. (Eds.) (Cham: Springer International Publishing, 2022), 530-539, https://doi.org/10.1007/978-3-031-15559-8_38.
7. International Organization for Standardization, "Information Technology—Process Assessment—Part 5: An Exemplar Software Life Cycle Process Assessment Model," ISO/IEC Standard 15504-5, 2012.
8. Society of Automotive Engineers, "Cybersecurity Guidebook for Cyber-Physical Vehicle Systems," SAE Standard J3061_202112, 2021.
9. United Nations Economic and Social Council—Economic Commission for Europe, "Software Update and Software Update Management System," Regulation 156, Brussels, 2021.
10. United Nations Economic and Social Council—Economic Commission for Europe, "UN Regulation on Uniform Provisions Concerning the Approval of Vehicles With Regard to Cyber Security and of Their Cybersecurity Management Systems," Technical Report ECE/TRANS/WP.29/2020/79, Brussels, 2020.
11. National Highway Traffic Safety Administration, "Cybersecurity Best Practices for the Safety of Modern Vehicles (Draft Update 2020)," Draft Update of DOT HS 812 333, Washington, DC, 2020.

Downloaded from SAE International by Tallinn University of Technology, Friday, June 13, 2025

12. British Standards Institution, "The Fundamental Principles of Automotive Cyber Security—Specification," BSI PAS 1885:2018, 2018.
13. Hu, S., Chen, Q.A., Sun, J., Feng, Y. et al., "Automated Discovery of Denial-of-Service Vulnerabilities in Connected Vehicle Protocols," in *30th USENIX Security Symposium (USENIX Security 21)*, USENIX Association, Vancouver, Canada, 2021, 3219-3236, https://www.usenix.org/conference/usenixsecurity21/presentation/hu-shengtuo.
14. Shen, J., Won, J.Y., Chen, Z., and Chen, Q.A., "Drift with Devil: Security of Multi-Sensor Fusion Based Localization in High-Level Autonomous Driving under GPS Spoofing," in *Proceedings of the 29th USENIX Security Symposium (2020)*, Boston, MA, 2020, 931-948.
15. Sun, J., Cao, Y., Chen, Q.A., and Morley Mao, Z., "Towards Robust LiDAR-Based Perception in Autonomous Driving: General Black-Box Adversarial Sensor Attack and Countermeasures," in *Proceedings of the 29th USENIX Security Symposium (2020)*, Boston, MA, 2020, 877-894, arXiv:2006.16974.
16. Kim, H., Ozgur Ozmen, M., Bianchi, A., Berkay Celik, Z. et al., "PGFUZZ: Policy-Guided Fuzzing for Robotic Vehicles," in *Network and Distributed System Security Symposium (NDSS)*, Virtual, 2021, 1-18, https://beerkay.github.io/papers/Berkay2021PGFuzzNDSS.pdf.
17. Kim, T., Kim, C.H., Rhee, J., Fei, F. et al., "RVFuzzer: Finding Input Validation Bugs in Robotic Vehicles through Control-Guided Testing," in *28th USENIX Security Symposium (USENIX Security 19)*, USENIX Association, Santa Clara, CA, 2019, 425-442, https://www.usenix.org/conference/usenixsecurity19/presentation/kim.
18. Vinzenz, N. and Oka, D.K., "Integrating Fuzz Testing into the Cybersecurity Validation Strategy," SAE Technical Paper 2021-01-0139 (2021), doi:https://doi.org/10.4271/2021-01-0139.
19. Ebrahimi, M. et al., "A Systematic Approach to Automotive Security," in *Formal Methods*, Lecture Notes in Computer Science, Chechik, M., Katoen, J.-P., and Leucker, M. (Eds.) (Cham: Springer International Publishing, 2023), 598-609, doi:10.1007/978-3-031-27481-7_34.
20. Oka, D., "Fuzz Testing Virtual ECUs as Part of the Continuous Security Testing Process," *SAE Int. J. Transp. Cyber. & Privacy* 2, no. 2 (2020): 159-168, doi:https://doi.org/10.4271/11-02-02-0014.
21. United Nations Economic and Social Council—Economic Commission for Europe, "Agreement Concerning the Adoption of Harmonized Technical United Nations Regulations for Wheeled Vehicles, Equipment and Parts which can be Fitted and/or be Used on Wheeled Vehicles and the Conditions for Reciprocal Recognition of Approvals Granted on the Basis of these United Nations Regulations," ECE/TRANS/WP.29/343/Rev. 30, 2022, 43.
22. The Ministry of Industry and Information Technology of China (MIIT), "Opinions of the Ministry of Industry and Information Technology on Strengthening the Management of Smart Connected Automobile Manufactures and Product Permit," MIIT Equipment Industry 103, 2021.
23. The Ministry of Industry and Information Technology of China (MIIT), "Suggestions on Strengthening the Type Approval Management of Intelligent & Connected Vehicle Manufacturers and Products," MIIT 5, 2021.
24. National Technical Committee of Auto Standardization, "General Technical Requirements for Vehicle Cybersecurity," GB/T 40861–2021, 2021.
25. The Ministry of Industry and Information Technology of China (MIIT), "Notice of the Ministry of Industry and Information Technology on Strengthening the Cyber Security and Data Security of Internet of Vehicles," MIIT Cybersecurity 134, 2021.
26. The Ministry of Industry and Information Technology of China (MIIT), "Security Technical Requirements for Connected Vehicle Based on Public Telecommunication Network," YD/T 3737-2020, 2020.
27. Chinese National Information Security Standardization Technical Committee, "Information Security Technology—Cybersecurity Technical Requirements for In-Vehicle Network Equipment," Technical Report, 2020.
28. Chinese National Information Security Standardization Technical Committee, "Technical Requirements for Cybersecurity of Electric Vehicles Charging System (Draft for Comments)," GB/T, 2020.
29. Chinese National Automotive Standardization Technical Committee, "Technical Requirements and Test Methods for Cybersecurity of Remote Service and Management System for Electric Vehicles," GB/T, 2021.
30. Japanese Ministry of Land, Infrastructure, Transport and Tourism Automobile Bureau, "Safety Technical Guidelines for Self-Driving Vehicles," Technical Report, 2018.
31. Information-Technology Promotion Agency, Japan, "Approaches for Vehicle Information Security," Technical Report, 2013.
32. Japan Automotive Software Platform and Architecture (JASPAR), "About Us," accessed November 10, 2023, https://www.jaspar.jp/en/about_us.
33. Ministry of Land, Infrastructure and Transportation, "Approach of Republic of Korea Harmonizing the UN Regulation No. 155," Technical Report, 2021.
34. ATIC, "Brief Analysis of the July 2022 Korean Regulatory Updates," Technical Report, 2022.
35. National Highway Traffic Safety Administration, "Understanding NHTSA's Regulatory Tools," Report, Washington, DC, 2017.
36. NHTSA, "Vehicle Cybersecurity," accessed June 28, 2023, https://www.nhtsa.gov/technology-innovation/vehicle-cybersecurity.
37. National Highway Traffic Safety Administration, "Report to Congress: 'Electronic Systems Performance in Passenger Motor Vehicles'," Technical Report, 2015.
38. McCarthy, C., Harnett, K., Carter, A., and Hatipoglu, C., "Assessment of the Information Sharing and Analysis Center Model," Technical Report DOT HS 812 076, National Highway Traffic Safety Administration, Washington, DC, 2014.

Downloaded from SAE International by Tallinn University of Technology, Friday, June 13, 2025

39. NHTSA, "NHTSA Seeks Comment on Cybersecurity Best Practices for the Safety of Modern Vehicles," accessed June 28, 2023, https://www.nhtsa.gov/press-releases/nhtsa-seeks-comment-cybersecurity-best-practices-safety-modern-vehicles.
40. National Highway Traffic Safety Administration, "Cybersecurity Best Practices for Modern Vehicles," Technical Report DOT HS 812 333, Washington, DC, 2016.
41. National Highway Traffic Safety Administration, "Cybersecurity Best Practices for the Safety of Modern Vehicles," Pre-Final, Washington, DC, 2022.
42. National Institute of Standards and Technology, "Framework for Improving Critical Infrastructure Cybersecurity," Technical Report, Gaithersburg, MD, 2018.
43. Automotive Information Sharing and Analysis Center, "Best Practices," Technical Report, 2016.
44. Automotive Information Sharing and Analysis Center, "Best Practices—Security Development Lifecycle," Technical Report, 2020.
45. Transport Canada, "Canada's Vehicle Cyber Security Guidance," Technical Report T46-61/2020E, 2020.
46. European Union, "The EU Cybersecurity Act," Technical Report, 2020.
47. European Union, "Cybersecurity Resilience Act," Technical Report, 2022.
48. Kraftfahrt-Bundesamt, "Application of the Rules for Designation/Recognition for Technical Services (Categories A, B, D)," Technical Report, 2021.
49. VDA QMC Project Group 13, "Automotive SPICE—Process Reference and Assessment Model for Cybersecurity Engineering," Core Specification 1.0, Quality Management Center of the German Association of the Automotive Industry, 2021.
50. Ministère de la Transition écologique et solidaire, "Cybersecurity in France for Civil Aviation," Technical Report, Direction générale de l'Aviation civile, 2018.
51. Agence nationale de la sécurité des systèmes d' information, "Cybersecurity Act," accessed November 10, 2023, https://www.ssi.gouv.fr/administration/reglementation/cybersecurity-act/.
52. Rocchi, J.-F., Bodino, P., De Tréglodé, H., Flury-Hérard, B. et al., "L'automatisation Des Véhicules; Annexe No. 9: La Cyber Sécurité. Inspection Generale de l'administration 16040-R," Inspection generale de l'administration and Conseil general de l'environnement et du developpement durable, 2017.
53. Agence nationale de la sécurité des systèmes d'information, "Analyse Des Contributions Reçues Suite à l'appel à Manifestation d'intérêt Sur La Certification de Sécurité de Niveaux Substantiel et Élémentaire," Technical Report, 2019.
54. United Kingdom Department for Transport, "The Key Principles of Cyber Security for Connected and Automated Vehicles," Technical Report, 2017.
55. British Standards Institution, "Automotive Cybersecurity Insights Paper," BSI PAS, 2021.
56. "Centre for Connected and Autonomous Vehicles, "Connected and Automated Vehicles: Process for Assuring Safety and Security (CAVPASS)," accessed June 28, 2023, https://www.gov.uk/guidance/connected-and-automated-vehicles-process-for-assuring-safety-and-security-cavpass.
57. Zenzic, "Cyber Resilience in Connected and Automated Mobility (CAM)—Cyber Feasibility Report," 2020.
58. Li, J., Zhao, B., and Zhang, C., "Fuzzing: A Survey," *Cybersecurity* 1, no. 1 (2018): 6, doi:https://doi.org/10.1186/s42400-018-0002-y.
59. Luo, F., Zhang, X., Yang, Z., Jiang, Y. et al., "Cybersecurity Testing for Automotive Domain: A Survey," *Sensors* 22, no. 23 (2022): 9211.
60. Mahmood, S., Nguyen, H.N., and Shaikh, S.A., "Automotive Cybersecurity Testing: Survey of Testbeds and Methods," in: *Digital Transformation, Cyber Security and Resilience of Modern Societies*, Studies in Big Data, vol. 84, Tagarev, T., Atanassov, K.T., Kharchenko, V., and Kacprzyk, J. (Eds.) (2021), Springer, Cham, https://doi.org/10.1007/978-3-030-65722-2_14.
61. Felderer, M., Zech, P., Breu, R., Büchler, M. et al., "Model-Based Security Testing: A Taxonomy and Systematic Classification," *Software Testing Verification and Reliability* 26, no. 2 (2015): 119-148, doi:10.1002/stvr.1580.
62. Marksteiner, S., Bronfman, S., Wolf, M., and Lazebnik, E., "Using Cyber Digital Twins for Automated Automotive Cybersecurity Testing," in *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*, Vienna, Austria, 2021, 123-128, doi:https://doi.org/10.1109/EuroSPW54576.2021.00020.

© 2024 International Alliance for Mobility Testing and Standardization (IAMTS). Published by SAE International. This Open Access article is published under the terms of the Creative Commons Attribution Non-Commercial, No Derivatives License (http://creativecommons.org/licenses/by-nc-nd/4.0/), which permits use, distribution, and reproduction in any medium, provided that the use is non-commercial, that no modifications or adaptations are made, and that the original author(s) and the source are credited.

Positions and opinions advanced in this work are those of the author(s) and not necessarily those of SAE International. Responsibility for the content of the work lies solely with the author(s).

Appendix IX

Paper IX

M. Malayjerdi, **A. Roberts**, O. M. Maennel, and E. Malayjerdi. Combined safety and cybersecurity testing methodology for autonomous driving algorithms. Proceedings of the 6th ACM Computer Science in Cars Symposium, pages 1–10, 2022.

Combined Safety and Cybersecurity Testing Methodology for Autonomous Driving Algorithms

Mohsen Malayjerdi
Department of Mechanical and Industrial Engineering
Tallinn University of Technology
Tallinn, Estonia
mohsen.malayjerdi@taltech.ee

Andrew Roberts
FinEst Centre for Smart Cities
Tallinn University of Technology
Tallinn, Estonia
andrew.roberts@taltech.ee

Olaf Maennel
Centre for Digital Forensics and Cybersecurity
Tallinn University of Technology
Tallinn, Estonia
olaf.maennel@taltech.ee

Ehsan Malayjerdi
Department of Mechanical and Industrial Engineering,
Tallinn University of Technology
Tallinn, Estonia
ehsan.malayjerdi@taltech.ee

ABSTRACT

Combined safety and cybersecurity testing are critical for assessing the reliability and optimisation of autonomous driving (AD) algorithms. However, safety and cybersecurity testing is often conducted in isolation, leading to a lack of evaluation of the complex system-of-system interactions which impact the reliability and optimisation of the AD algorithm. Concurrently, practical limitations of testing include resource usage and time. This paper proposes a methodology for combined safety and cybersecurity testing and applies it to a real-world AV shuttle using digital twin, software-in-the-loop (SiL) simulation and a real-world Autonomous Vehicle (AV) test environment. The results of the safety and cybersecurity tests and feedback from the AD algorithm designers demonstrate that the methodology developed is useful for assessing the reliability and optimisation of an AD algorithm in the development phase. Furthermore, from the observed system-of-system interactions, key relationships such as speed and attack parameters can be used to optimise testing.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

KEYWORDS

automotive cybersecurity, safety testing, autonomous driving

ACM Reference Format:
Mohsen Malayjerdi, Andrew Roberts, Olaf Maennel, and Ehsan Malayjerdi. 2022. Combined Safety and Cybersecurity Testing Methodology for Autonomous Driving Algorithms. In *Computer Science in Cars Symposium (CSCS '22), December 8, 2022, Ingolstadt, Germany*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3568160.3570235

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CSCS '22, December 8, 2022, Ingolstadt, Germany
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9786-5/22/12…$15.00
https://doi.org/10.1145/3568160.3570235

1 INTRODUCTION

Testing autonomous driving (AD) algorithms for performance under safety test cases is a predominant focus for developers to assess the reliability of the algorithm and for optimisation. AD algorithms are also susceptible to manipulation from cyber threats which target the advanced hardware technologies sensor telemetry which serves as an essential input for perception, detection, and control decisions [2, 12, 20]. Existing methods [3, 8] for testing are challenged by the complexity of evaluating system-of-system interactions to identify key relationships and parameters, and limitations of testing inherent to real-world AV programs, resource usage and time. The main idea of this paper is to establish a method for combined safety and cybersecurity testing of developmental AD algorithms to evaluate system-of-system interactions to identify and investigate parameters that impact safety and the effect of cyber attacks, and to develop future ideas for optimisation of testing. To this end, the paper focuses on three research questions aligned with the challenges of combined safety and cybersecurity for AD algorithms.

RQ1 How can AD algorithm designers evaluate the reliability and optimisation of the AD algorithm to both safety and cybersecurity test cases?
RQ2 Cybersecurity testing is predominantly conducted on well-established AD algorithms. How can combined safety and cybersecurity testing be conducted on a developing AD algorithm?
RQ3 What key relations and parameters can we identify that can optimise safety and cybersecurity testing?

To evaluate these research questions, we apply our methodology to a developing AD algorithm in a digital twin, software-in-the-loop (SiL) simulator and real-world AV testing environment. Cybersecurity testing and safety testing are often conducted separately, reducing our understanding of the relationship between failures of the algorithm caused under normal safety scenarios and failures caused by the impact of cyber attacks. For AD algorithms in the development stage, where the reliability and optimisation of the AD algorithm to safety scenarios have not been established, this exploration of the relationship between safety and cybersecurity can offer novel insights to improve the awareness of the AD algorithm designer to shortcomings in the algorithm.

The major contributions of this paper are the following:

- Methodology for combined safety and cybersecurity testing
- Safety and cybersecurity test cases conducted on an AD algorithm under development, and with feedback from the AD algorithm designer
- An analysis of the combined safety and cybersecurity test cases that identifies key relations and the sensitivity of parameters.
- All the code, our AV simulation configurations and research data used in the combined safety and security testing will be available for the research community on GitHub.

2 TARGET SYSTEM

2.1 Low-Speed AV Shuttle for Public Transportation

The target AV for this study, iseAuto (see Fig. 1), is a real-world AV shuttle for public transportation, operating in numerous EU countries.

Figure 1: iseAuto autonomous shuttle

The shuttle was developed as part of a project at Tallinn University of Technology's AV research group. The objective of this project is to build an open-source AV shuttle that provides a smart city test bed within the university campus, enabling different types of urban mobility research. Currently, this SAE level 4 and 5 shuttle is operating on the campus for experimental and study purposes. iseAuto uses a multi-LiDAR sensor system for perception and localisation. Two Velodyne LiDARs are mounted at the top front (VLP-32) and the back (VLP-16) of the vehicle, in addition to two Robosense RS-Bpearl at both sides (left and right), to decrease the sensor blind zone around the car.

2.2 Autonomous Driving Algorithm

The AV uses Autoware.ai [11] autonomous software stack which is an open-source AD software. This software enables us to employ different algorithms for each main part of the autonomous system including localization, sensing, detection, and navigation. Open-Planner navigation planning algorithm.

In this study, we focused on OpenPlanner as one of the most widely used path-planner modules in the AD software. In the latest version of this algorithm, which is currently 2.5, the module has become noticeably more advanced in terms of supporting various high-definition map formats, predicting the trajectories of other actors, and using a kinematics-based trajectory generator [5]. This version is compatible with Autoware.ai 1.15. Open-planner combines global and local planners that jointly utilize the road network map to generate local waypoints based on a global route and manage discrete behaviours such as avoiding dynamic obstacles and following traffic lights.

The local planner module generates tracks parallel to the main path defined by the global planner. These tracks are named rollouts (see Fig. 2). The trajectory evaluator assesses all possible rollouts in case an obstacle blocks the path. Then, the behaviour selector will lead the AV to the new safe rollout. Figure 2 shows how open-planner selected rollout number 6 in order to pass the non-player character (NPC). It also detects the curb lines and avoids those rollouts which intersect the curbs.

The algorithm uses the output of the kf_contour_track algorithms to consider all the perceived objects based on the LiDARs point cloud in its local path planning. Earlier, the euclidean clustering algorithm received the filtered point cloud data and prepared point clusters, which is the input of the kf_contour_track. This combination of cluster and contour tracking is done in each sequence for the open-planner to evaluate possible trajectories and create the behaviour based on that. Figure 3 shows the diagram of how the open-planner module works under the AD software package.

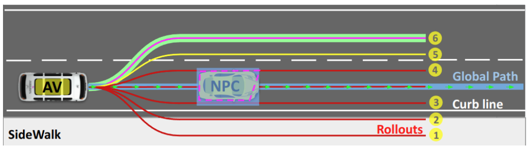


Figure 2: How open-planner generates different trajectory to pass an object

3 COMBINED SAFETY AND CYBERSECURITY TESTING METHODOLOGY FOR AD ALGORITHMS

The architecture of the proposed combined testing methodology is presented in Figure 3. This method takes advantage of a high-fidelity software in the loop (SiL) simulation [16] approach to validate and verify the performance of a AD software under critical cyber security conditions. This method consists of three main following elements:

- Attack script: which simulates a critical security condition.
- High-fidelity simulator: It is a game engine environment that provides the physics for modeling sensors and motion.
- AD software: It is the autonomous driving software that controls the AV.

The combined safety and cybersecurity methodology consisted of the following iterative steps:

- **Scenario Selection**
- **Analysis of the scenario to extrapolate the safety evaluation criterion applicable**
- **Safety Test Case Setup**
  - Initialisation of the SiL high-fidelity simulator and configuration to the real-world AV

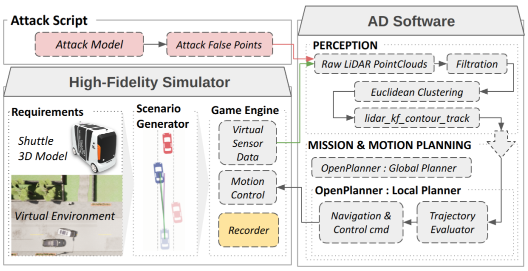


Figure 3: Architecture of the testing platform

- Initial scenario testing using the safety test cases to assess the reliability of the algorithm and the quality of the test data
- Optimisation of the safety test cases to select a subset of the scenario tests to assess the reliability of the algorithm
- Run of the safety test case scenarios
- Selection of distinct safety test case scenarios which provide most stable results in terms of success of mission and safety violation

- **Cybersecurity Test Case Setup**
  - Analysis of the scenario to determine cyber attack strategy for test cases
  - Development of the code for adversary generation in the SITL high-fidelity simulator
  - Selection of attack parameters
  - Optimised the cybersecurity test cases
  - Evaluate cybersecurity test cases in SiL high-fidelity simulator
  - Real-World AV Testing for safety and cybersecurity
- **Results Analysis**
  - Analysis of the performance of AD algorithm to safety criteria
  - Analysis of sensitivity of attack parameters and driving parameters

3.1 Testing Environment

All tests are conducted in a virtual environment powered by the "Unreal game engine" (Unreal) [4]. Carla simulator [6] is one of the open-source high-fidelity vehicle simulators capable of connecting to different AD software and scenario generator applications. In this study, we use Carla 0.9.13 as the high-fidelity simulator. Figure 3 illustrates the requirements for the high-fidelity simulator to conduct simulation testing which are two components, the digital twin of our AV and the virtual replication of our target environment. These replicated components help us to gain more accurate results of the proposed platform [14]. The AV digital twin is a 3D model of our real-world world AV shuttle, designed in Blender, a graphical 3d modelling software, and imported and built in Unreal for deployment in Carla. This model uses the same dimension and sensor configuration (model, position, and orientation) from the real AV shuttle. The environment digital twin, in our case, is identical to the location where we are testing and operating our shuttle, this includes the urban details and vegetation. The next module in the simulator is a scenario generator that produces the desired scenario based on the user input specification. Finally, the simulator engine generates sensor data from sensors, including LiDARs, cameras and others and publishes it for other blocks (see Fig. 3 the simulator block). Then, the AD software receives this data as raw LiDAR point-cloud information and processes the data as mentioned in the diagram (Figure 3).

This simulation setup was implemented on a desktop computer with the following configuration:

- Intel® Core™ i7-11700K @ 3.60GHz × 16 cores
- NVIDIA GeForce RTX 3080 10 GB
- RAM: 128 GB

3.2 Scenario Selection

To evaluate the combined safety and cybersecurity testing, we chose a simple overtaking maneuver, which is one of the most safety challenging operations [13]. Figure 4 shows the functional level of the planned scenario. To generate a variety of distinct scenarios, we opt for the initial relative distance to the NPC D_x and the NPC constant speed S_{NPC} as the distinct scenario parameters.

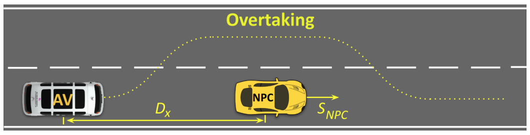


Figure 4: Dx and S_{NPC}, define the initial relative distance to the NPC and the constant NPC speed in each scenario

Table 1: Target scenarios definition

Actor	Speed	D_x	Goal
AV	$[0{:}6] m/s$	0 (m)	overtake the NPC safely
NPC	[1 1.4 1.8 2.1 2.5]	[15 20 25](m)	keep moving

3.3 Safety Evaluation Criteria

In determining the evaluation criteria for AV safety we considered two conditions, 1) mission success and 2) safety violations. A safety violation consists of a collision and dangerous driving behaviour. In determining which criteria to apply, we considered the EuroNCAP [1] and ISO26262 [10] standards as well those used in composite studies [3, 7, 8]. We derived that the safety goal of the AD algorithm is to execute the overtaking mission without colliding or interfering with other ego vehicles or objects and without exhibiting driving behaviour which is dangerous to the AV passengers. Table 2 details the safety criteria applied in our experiments.

3.4 Safety Test Case Setup

To evaluate the reliability and optimisation of the AD algorithm for the overtaking manoeuvre, we, firstly, initiated a run of 50 distinct scenarios in the high-fidelity simulator, repeating 6 times. Each scenario was repeated 6 times to ensure the reproducibility

Table 2: Safety Evaluation Criteria

Safety Condition	Data Label	Description	Metric
Succeed	Suce	AV Successful complete the mission	Pass/Fail
Not Finished	NotF	Failure to finish the mission	Pass/Fail
Distance-to -Collision	DTC	Violation of the safe distance between AV and NPC	AV within 0.5m of other vehicle
Break on Driving Lane	BrD	AV initiates emergency break on driving lane	Pass/Fail
Break on Passing Lane	BrP	AV initiates emergency break on passing lane	Pass/Fail
Collision	Col	AV collides with NPC	Pass/Fail
Violation	V	Safety Violation	

of the outcome. With the mentioned desktop configuration, it took approximately 100 *sec* for each scenario and, in total, 8.3 hours for 300 runs. The purpose of the first scenario run was to provide a general overview of the performance of the algorithm. We targeted a range of 1 to 3 m/s for the NPC speed and 15 to 30 m for the initial relative distance to the NPC for selecting the 50 distinct scenario parameters. The results showed that the AD algorithm could not safely overtake the NPC at an NPC speed higher than 2.5 m/s and a distance (D_x) of more than 25 m.

Although a high number of scenario variations shows better coverage in the scenario space to find corner cases, it will lead to an increase in the time duration of the runs. Furthermore, the number of each scenario repetitions was not sufficient to statistically explain the occurrence of each safety violation. Finally, it is worth mentioning that, as our primary study focus is not just the validation of the AV performance, we need to use an optimum number of trials for both safety and cyber test cases. Due to this, we limited the scenario parameters space to the intervals listed in Table 1 that regressed the test set to 15 distinct cases in a full factorial setup. This enabled us to repeat the simulation of these test cases 50 times and apply the full set of safety criteria: collision, DTC, break in passing lane, break in driving lane, failure to finish, and mission success.

Each scenario is generated by the Carla scenario runner utilizing the Python behaviour trees to handle series and parallel events in the scenario. Figure 5 depicts the scenario scheme starting with the main sequence behaviour. This series begins with transforming the actors into the environment and finishes by destroying the actor block. A parallel behaviour (Driving Toward Intersection) is defined to run the attack and the scenario stop block while the NPC follows the defined waypoint. For safety test case scenarios, the attack block is skipped, and the scenario waits till the stop criteria are satisfied.

3.5 Cyber Test Case Setup

To determine the cyber attack strategy for implementation in this test scenario, we analysed the overtaking scenario and its applicability to state-of-the-art attacks on AD algorithms. We selected

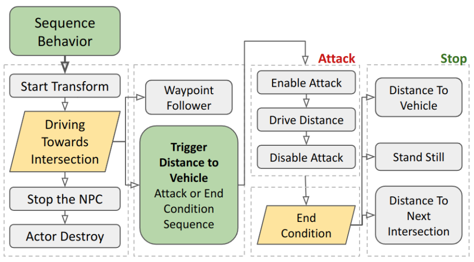


Figure 5: Flow-graph of how each scenario is processed in the simulation platform

LiDAR spoofing as it is a realistic attack in the driving environment of our real-world AV shuttle [3] and its impact is relevant to safety outcomes due to the likelihood that the manipulated driving behaviour will result in collisions, emergency breaking, and lane violations [20]. Attacks on LiDAR perception predominantly focus on spoofing LiDAR 3D point-clouds through the following means: 1) injection of adversarial LiDAR 3D point cloud data to add adversarial objects to the driving environment inducing a *false positive result* of the AD perception [3, 17] 2) removal of LiDAR 3D point cloud data to perturb the ability of the perception algorithm to detect objects in the driving environment, also known as a *false negative result* [8, 9] 3) manipulating LiDAR 3D point cloud data to obfuscate the true distance of environmental objects (Other road vehicles, pedestrians, other road objects) from the AV, causing the perception to *fail translation* 4) implementation of adversarial mesh in the driving environment to introduce manipulated points into the LiDAR 3D point cloud and create unpredictable perception events [19]. The aim of the attacker, in adversarial LiDAR threat models, is to induce the victim AV to perform dangerous driving maneuvers, which include; emergency breaking, collisions, and exceeding the limits of the driving lanes. Variables that have been shown to influence attack success include; angle of attack of the adversarial point cloud vector, density of the spoofed points, duration of the broadcast of spoofed points, distance of the point cloud to the target [3, 8, 17, 20]. We implemented a variation of the attack suggested by Yang et al. [20], where the adversary creates an adversarial roadside object to inject spoofed, malicious LiDAR point clouds into the target AV LiDAR. In our attack, an adversary has configured a LiDAR on the roadside to inject malicious point cloud data into the AV as it is conducting the overtaking manoeuvre. Figure 6 demonstrates the implementation of our attack.

Using the knowledge gained from literature [8, 17, 20], the parameters we chose to generate our attack are: density of the LiDAR point clouds, frequency (the publishing rate of the fake points), duration of the adversarial point cloud broadcast, and location, which is the relative location between the target vehicle and NPC. As an infinite number in the range of each of the parameters can be chosen, we decided to limit our testing to parameter values that had demonstrated utility to investigate the impact of cyberattacks on AD algorithms. For example, Hallyburton et al. [8] found that the success of cyber attacks increased when spoofed point density were over 80. Therefore we chose a range for spoof point density from 50 to 300.

3.5.1 Taguchi Analysis. In this study, we use the Taguchi method for statistical evaluation [18] of the attack parameters effect on each safety criterion. The number of tests with four parameters and 3 levels for each in full factorial mode would become unrealistic to perform, noting that each experiment should repeat 50 times (81x50 = 4050 distinct scenarios). A design of the experiment is recommended in order to avoid full factorial tests and reduce the number of tests without compromising accuracy [18].

A Taguchi design of experiment (DOE) technique [18] was applied to quantify the influence of four proposed attack parameters; the false points (FP) density, the FP frequency, the attack duration, and the attack location. In total, 9 experiments were designed with 3 different values for the four parameters. The analyses hence possess four factors and three levels for the Taguchi L9 matrix. Table 3 lists the configuration for each run conducted for cybersecurity tests.

Table 3: Taguchi L'9 matrix for study of factor influence

Num.	*Density*	*Frequency*	*Duration*	*Location*
1	50	5	3	3
2	50	7	6	6
3	50	10	9	9
4	150	5	6	9
5	150	7	9	3
6	150	10	3	6
7	300	5	9	6
8	300	7	3	9
9	300	10	6	3
	[50 150 300]	[5 7 10]	[3 6 9]	[3 6 9]

Figure 6 demonstrates the cyber attack setup within the overtaking scenario (Please note, the Figure only depicts the overtaking frame and not the entire overtaking sequence.). The proposed attack model will start by generating spoof points from the designated place on the roadside. At the starting point, P_1, the AV has relative distance to NPC that defines the attack location. After a specific duration (Attack Duration), the AV reaches, P_2. While the attacker keeps the malicious LiDAR pointing toward the AVs front LiDAR. Overall, the spoofed point direction changes from θ_1 to θ_2.

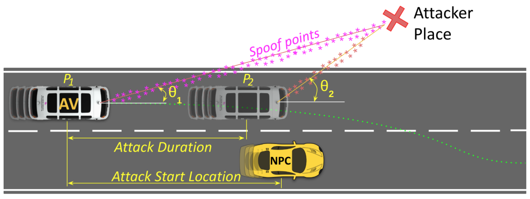


Figure 6: Attack scheme

Code was created for the generation of the adversarial LiDAR fake points to be run in the digital twin, high-fidelity simulation environment. This is available on the GitHub site [15].

4 RESULTS AND ANALYSIS

In this section, we present the results of the safety and cybersecurity testing of the end-to-end AD algorithm. The purpose of the safety test case results is to evaluate the reliability and optimisation of the algorithm.

4.1 Safety Test Case

The aim of the testing is to assess the utility of the methodology to evaluate the relationship between the reliability of the AD algorithm to safety and the impact of cybersecurity. As the testing is based on a real-world AV, we were motivated to establish what results could be gained from an amount of tests that took into account the requirements for CPU and GPU resources and the time involved in running high-fidelity simulations. For instance, 50 distinct scenarios run 3 times expends x amount of resources, and takes x amount of time. Therefore, we, firstly, performed a baseline evaluation test where we ran 50 distinct scenarios of the overtaking manoeuvre, 3 times. Each scenario is distinct based on changes to parameters such as NPC speed and initial distance to NPC.

In our proposed simulation platform, we perform 15 distinct scenarios, run 50 times; in total, 750 consecutive simulation runs were conducted. Table 4 shows the parameters of the distinct scenarios evaluated against the safety criteria. Using our configuration for testing, the AD algorithm shows the performance for the overtaking manoeuvre with a success rate of 43.9% of the simulated scenarios, whilst, 66.1% are safety violations.

In Figure 7 is the performance of the AD algorithm.

Table 4: Summary of the safety simulation

	D_x	S_{NPC}	V_{Col}	V_{DTC}	V_{BrP}	V_{BrD}	V_{NotF}	V_{Suce}
1	15	1	18%	22%	0%	10%	24%	26%
2	20	1	18%	40%	8%	6%	18%	10%
3	25	1	4%	20%	32%	8%	20%	16%
4	15	1.4	6%	32%	16%	2%	12%	32%
5	20	1.4	22%	26%	14%	6%	2%	30%
6	25	1.4	4%	12%	22%	8%	0%	54%
7	15	1.8	36%	34%	8%	2%	6%	14%
8	20	1.8	22%	12%	2%	2%	0%	62%
9	25	1.8	18%	6%	0%	4%	0%	72%
10	15	2.1	4%	0%	4%	2%	4%	86%
11	20	2.1	8%	10%	0%	0%	0%	82%
12	25	2.1	24%	0%	0%	4%	0%	72%
13	15	2.5	14%	6%	0%	6%	2%	72%
14	20	2.5	44%	22%	14%	0%	2%	18%
15	25	2.5	64%	18%	0%	0%	6%	12%
		mean	20.4%	17.3%	8.0%	4.0%	6.4%	43.9%
		STD	16.8%	2.3%	9.8%	3.2%	8.1%	28.3%
		min	4%	0%	0%	0%	0%	10%
		max	64%	40%	32%	10%	24%	86%

NPC speed is an important parameter as it influences the decision control for the critical cut-in manoeuvre of the overtaking mission. In the context of the results of the simulations, we can see that NPC speed impacts certain safety criteria.

The first such relation that can be seen, is that more collisions are caused at high speeds, > 2.1 m/s. This can be the effect of a poor trajectory evaluator that doesn't consider the prediction of the other actors motions in the process of the waypoint generation. In

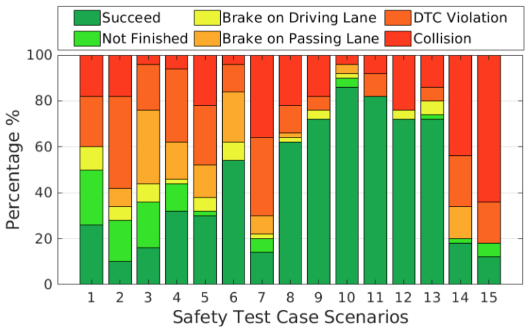


Figure 7: The 15 distinct scenarios

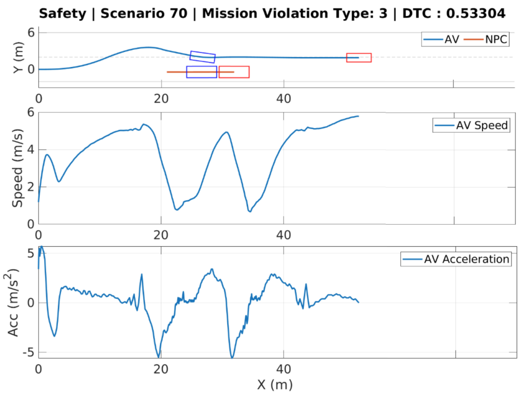


Figure 8: A Brake on Passing Lane safety violation

most collision cases the AV tried to perform a cut-in while the NPC collided from the right side. The probability of this safety violation will be increased as the NPC speed increases.

NPC speed also impacts the likelihood of a DTC safety violation. In the range of the NPC speed parameter, 1 m/s to 1.8 m/s, it can be observed that AV Shuttle violates the safe distance to the NPC. This can be due to the AV speed adjusting relative to the NPC speed and the cut-in is attempted at low-speed, whilst acceleration is required to safely attempt the cut-in. This low-speed cut-in firstly causes a DTC violation and if the overtaking manoeuvre progresses it causes a collision. DTC and collision correlate based on the relative speed. A low-speed NPC will likely result in a DTC violation, whilst in a higher-speed scenario, a collision is more likely to happen.

In the lowest speed range, 1 m/s to 1.4 m/s, it is more likely that the AV will initiate an emergency break in the passing lane. This is due to the relationship of the NPC speed to the AV Shuttle speed. The emergency break on the passing lane at low speeds is caused by a failure of the open-planner trajectory evaluator to effectively plan the overtaking trajectory. Figure 8 demonstrates the AV emergency break in the passing lane, for a scenario with an NPC Speed of 1 m/s. The upper rectangle represents the AV and the lower rectangle is the NPC. The two rectangles closest to the left represent the frame that the first emergency break on the passing lane safety violation occurs. The most right rectangles represent the end of the mission. The AV speed and the acceleration verify two hard brakes in the mission while it was in the passing lane. The failure of the trajectory planning of the open-planner algorithm is apparent.

The failure to finish the overtaking mission is most prominent at the lowest speed, 1 m/s, this is due to the time the AV Shuttle is taking to perform the cut-in process and therefore cannot enact the overtaking manoeuvre within the simulation timeout which is 40 s. It was observed that for the proposed configuration, for the lower speed of the NPC, the open-planner trajectory evaluator is not reliable as it suggests waypoints that are not within safe navigation and this is due to the lack of firm decision-making of which roll-out to choose. Ultimately, this causes collision and DTC safety violations. Furthermore, the failure to finish the simulation results, we see the low-speed delays in the overtaking manoeuvre decision making which results in the breach of the 40 s time-out.

The success rate of the safety test cases increases as the NPC drives from 1.4 to 2.1 m/s speed. This focal success point around scenario 10 with an NPC speed of 2.1 m/s can be a sign of matching the current configuration of perception and open-planner with the scenario situation.

The safety metrics results are shown in Figure 10 based on the initial relative distance from the AV to NPC. It shows that the rate of collision safety violations for longer initial distances from NPC slightly increased while the success rate decreased. This is the only trend that can be identified from results for initial relative distance, so it can be concluded that speed is a more determining parameter for the safety testing of our AV.

Overall, the results in Figure 7 indicate that speed is a critical parameter for our AV safety testing platform.

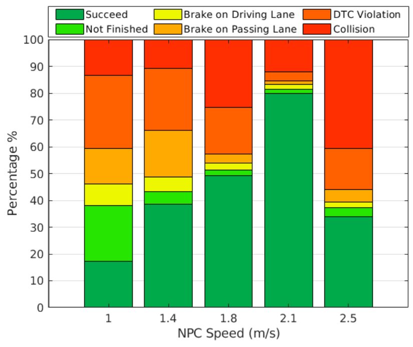


Figure 9: Test Results based on NPC Speed

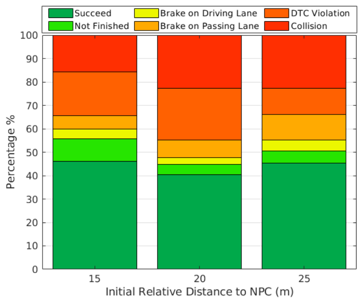


Figure 10: Results based on Initial Relative Distance to NPC

4.2 Cybersecurity Test Case

For the cybersecurity test cases we chose 2 of the 15 distinct scenarios (Figure 7). This was to allow a greater scale of testing to be conducted on a select number of relevant scenarios. Scenario 10 was chosen as it demonstrated the most reliable performance, in terms of the most successful overtaking manoeuvres. Scenario 2 was chosen as it demonstrated the least successful results for overtaking. These two scenarios were run 50 times each, as had been conducted with the safety scenario runs. Figure 11 shows the performance of cybersecurity testing, conducted on scenario 2 and scenario 10, in comparison to safety test cases.

Scenario 10 results reveal a discernible impact of the cyber attack. The LiDAR spoofing attack causes an increase in safety violations, prominently, in collisions and emergency breaking in the passing lane. This is also a concurrent result of the Scenario 2 test cases. Figure 3 shows the control level view, that incorporates sensor perception and mission and motion-planning. In the safety violation cases, we noticed that the euclidean clustering and kf_countour detect the spoofed LiDAR injection as an object and this false positive detection impacts the local-planning to force the AV to make the cut-in, in the overtaking manoeuvre process. Specifically, as the placement of the adversarial LiDAR device is on the left of the AV, the roll-outs of the left-side are blocked by the trajectory-evaluator. This forces the AV to veer right and attempt the cut-in process that causes predominantly collision, DTC safety violations.

Cao et al. [3] and Hallyburton et al. [8] identify density of the spoofed points to be one of the key variables affecting cyber attack success rate. Figure 12 and figure 13 present the sensitivity of each attack parameter according to the cyber attack test cases. From evaluating the raw data of the test sets, and the sensitivity analysis for the cyber attack test cases of scenario 10, we concur with these assessments. We find the rate of collisions is influenced by the density of the point cloud and the location of the attack. We can also see the influence the point of attack and duration have on causing a break on passing lane safety violation. As the duration of transmitting of the LiDAR point clouds increases and the location of the attack is further from the NPC, the likelihood of the AV initiating its breaks is higher.

In comparison, Scenario 2 cyber attack test case results show that safety violations are less sensitive to attack parameters. This can be due to the difficulty in interpreting the impact of cybersecurity on this scenario due to the already high rate of safety violations of the algorithms exhibited in the safety test case.

Table 5: Results of Cyber Attack applied to Scenario 10

Num.	V_{Col}	V_{DTC}	V_{BrP}	V_{BrD}	V_{NotF}	V_{Suce}
1	54%	20%	2%	0%	6%	18%
2	38%	38%	6%	2%	6%	10%
3	30%	28%	22%	2%	4%	14%
4	24%	28%	16%	6%	2%	24%
5	26%	16%	12%	6%	4%	36%
6	4%	4%	6%	4%	0%	82%
7	32%	14%	14%	6%	0%	34%
8	50%	24%	8%	2%	0%	16%
9	50%	30%	2%	2%	0%	16%
mean	34.2%	22.4%	9.8%	3.3%	2.4%	27.8%
std	15.9%	10.1%	6.7%	2.2%	2.6%	22.2%
min	4.0%	4.0%	2.0%	0.0%	0.0%	10.0%
max	54.0%	38.0%	22.0%	6.0%	6.0%	82.0%

Table 6: Results of Cyber Attack applied to Scenario 2

Num.	V_{Col}	V_{DTC}	V_{BrP}	V_{BrD}	V_{NotF}	V_{Suce}
1	16%	34%	28%	8%	14%	0%
2	26%	34%	20%	0%	8%	12%
3	20%	42%	20%	4%	6%	8%
4	26%	34%	16%	0%	14%	10%
5	22%	36%	16%	0%	20%	6%
6	22%	32%	20%	0%	18%	8%
7	0%	0%	0%	0%	0%	0%
8	0%	0%	0%	0%	0%	0%
9	0%	0%	0%	0%	0%	0%
mean	14.7%	23.6%	13.3%	1.3%	8.9%	4.9%
std	11.4%	17.9%	10.6%	2.8%	7.9%	4.9%
min	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
max	26.0%	42.0%	28.0%	8.0%	20.0%	12.0%

4.3 Real-World AV Testing

The real-world AV testing was conducted on a private road environment using our AV Shuttle, and an NPC vehicle (turquoise Mitsubishi iMIEV). The NPC vehicle is stationary during the tests as a safety assessment deemed it was too dangerous to conduct the experiment with a moving vehicle. This is due to the experiment being within a road environment where pedestrians and other vehicles are present. We conducted 3 test cases; a safety test case, cybersecurity test case and an optimised cybersecurity test case. The first test was an overtaking safety scenario. Two repetitions of the safety test case were conducted. The first test demonstrated a successful execution of the overtaking mission. The second test resulted in a DTC safety violation. The AV motioned to within 0.42 m of the NPC. The DTC violation is evident in Frame 3 of Figure 14, which details the second overtaking safety test case. Frame 4 demonstrates the eventual overtake after the DTC safety violation. Whilst the number of repetitions in the real-world pale in comparison to those conducted in the simulator, the real-world results

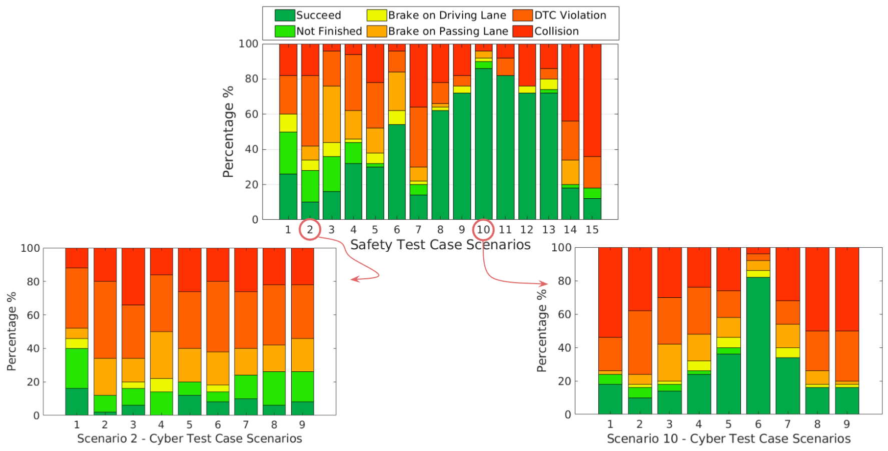


Figure 11: Performance Results Comparing Cyber Vs Safety Test Cases

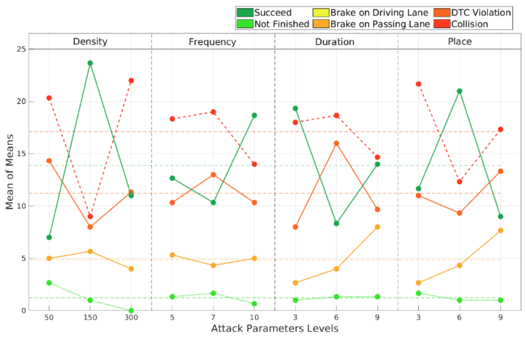


Figure 12: Scenario 10 - Cyber Attack Test Cases - Parameter Sensitivity

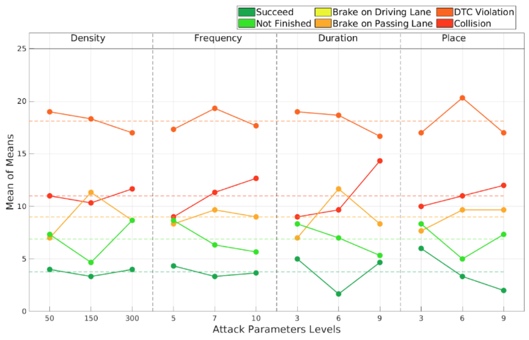


Figure 13: Scenario 2 - Cyber Attack Test Cases - Parameter Sensitivity

Table 7: Result of the 3 real-world test cases

Test Type	Num. of repeats	success	Safety Violations
Safety Tests	2	1	1 DTC=0.42m
Cyber Tests	2	1	1 DTC=0.38m
Optimised Cyber Tests	1	0	1 DTC=0.32m

concur with simulation results, that the AD algorithm does not have enough reliability for the deployment in real-world missions.

The cybersecurity test was conducted 3 times. Table 7 lists all the real-word experiments and their results. The first cybersecurity test demonstrated no impact from the spoofed LiDAR points and the overtaking manoeuvre was successful. The second cybersecurity test resulted in a DTC violation, the AV motioned to within 0.38 m of the NPC. After these two tests, we optimised the target angle of the spoofed points in relation to the attack scheme in Figure 6, to reduce the attack starting angle of θ_1. We did this because during the real-world test we observed that the reduced angle would provide assist the spoofed points to be closer to the AV trajectory and would cause the AV to detour from its intended route. It can be seen that this did work as the DTC decreased to 0.32 m. Figure 15 depicts the real-world cybersecurity test. Frame 2 represents the moment the attack was generated and perceived by the AD algorithm.

The videos and images related to the real-world tests are found on GitHub site.

5 DISCUSSION

From the analysis of the results we interpreted that different safety violations are connected to different modules of the AD algorithm.

Perception Module) We interpreted the cause of safety violations of the emergency break in the passing lane and emergency break in the driving lane to be related to the quality of the ground filtration. As we observed, ground filtering outcome changes during the AV maneuvers (turns) because the shuttle body is tilted because of suspension and this results in the lidar reference frame orientation

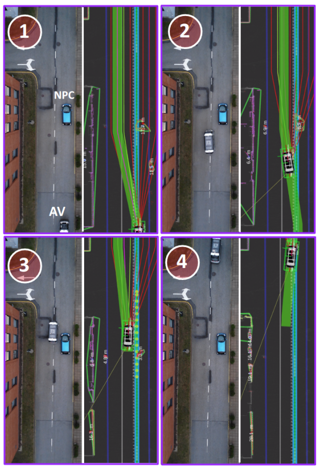


Figure 14: Real-World AV Test - Safety Test Case

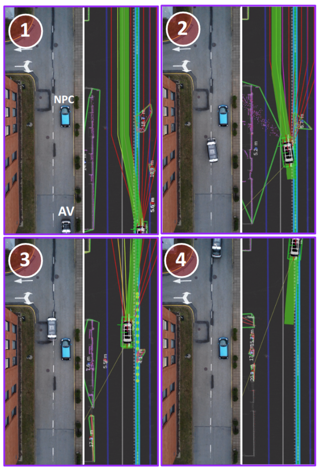


Figure 15: Real-World AV Test - Cyber Attack Test Case

changing. Then some part of the ground point cloud as an unfiltered perception can be seen in the detection algorithms as an obstacle. This fake sudden obstacle might stop the AV during the motion. The spoofed LiDAR point cloud threat model is likely to make this condition worse. Optimisations for this: New body designs to rectify or limit the issues of LiDAR with the physics of the AV Shuttle are being developed. To focus specifically on these corner and edge cases and look at optimisation of the filtering of the perception algorithm. The latter recommendation is complicated by the fact it may include trade-offs; if the LiDAR perception algorithm is specifically tuned for this corner/edge case it could lead to over-filtration in normal driving scenarios, therefore this is one of the optimisation options to resolve the perception for the algorithm.

Open-Planner Module) We interpret the cause of safety violations for DTC and collision as due to an issue of the open-planner in predicting the trajectory of the NPC during the process of performing a cut-in, in front of the NPC. The optimisation would involve incorporation of features that would enable the prediction of the trajectory of the NPC and for perception improve the perception of the side-lidar to accurately perceive the NPC. We found that optimising all the perception and open-planner parameters for our shuttle model would significantly improve the reliability of the AD algorithm.

5.1 Open-Planner Developer Feedback

We sent a presentation of our results to the developers of the open-planner AD algorithm. In response, they acknowledged that it is a developing algorithm and we are engaged in more detailed discussions with them on how to optimise the algorithm. They also announced they are transitioning from Autoware.ai to Autoware.universe which is a more developed and advanced platform. Amongst their responses, they also pointed to the novelty of receiving feedback on the reliability of cybersecurity test cases in addition to safety test cases.

6 RELATED WORK

The closest contributions to our work are Yang et al. [20], Hallyburton et al. [8], Cao et al. [3] and Zhu et al. [21]. Each of these papers utilises a LiDAR spoofing threat model that varies based on the method for delivering the attack, adversarial generation and the type AD algorithm. Hallyburton et al. [8] target camera and LiDAR sensor fusion. They identify a blind spot between the camera and LiDAR sensor at the rear of the target AV. They use a malicious, 3D LiDAR point cloud array to inject malicious spoof points into the rear angle of the target AV. The attack was tested in a high-fidelity simulation and real-world against multiple perception algorithms. The results revealed a high rate of success utilising this attack. Cao et al [3], Yang et al [20], and Zhu et al [21] developed LiDAR spoofing attacks based on a threat model of a malicious LiDAR 3D point cloud injection in the road environment and by the roadside. Each of these contributions demonstrated that cyber attack results from AV simulation testing can be used to identify key parameters such as point cloud density, attack location and duration and that these parameters can be optimised to test the robustness of perception algorithms. We chose to extend from the related literature, in our work, in three areas; simulation testing configuration, safety criteria evaluation and target AD algorithm is in the developmental phase and is used within a real-world AV program. A feature of the selected work is that simulation testing often selected only one frame or a limited amount of frames and therefore the full driving mission was not observed. Whilst this is useful for reducing testing resource usage, running massive scale of tests and applicable to the scope of their work, as our study evaluates the end-to-end AD algorithm and combines safety, our study focused on conducting simulation testing for the entire driving mission. Secondly, the evaluation of cyber attacks focused on attack success rate and attack parameters whilst the safety impact on the AV as a result of cyber attacks was not as clearly elaborated. In our study, we evaluate the cyber attack test cases with the same criteria as the safety case to derive the category of safety violation. Lastly, most of the simulations use default AV configurations and evaluate well-established algorithms. Our study uses a simulator configured for a real-world AV and evaluates an AD algorithm in the developmental stage where reliability and optimisation are required to be assessed under safety, non-cyber test cases before the impact of cyber attacks can be understood.

7 CONCLUSION

We developed a combined methodology for safety and cybersecurity utilising a digital twin, high-fidelity simulation environment and a real-world AV shuttle for public transportation. We evaluated our approach on a developing AD algorithm consisting of open-planner, as the mission and motion-planning module. We evaluated the reliability of the AD algorithm on an overtaking scenario using test cases for safety and cybersecurity based on a LiDAR spoofing attack. The combined safety and cybersecurity testing enabled us to assess the outcome of the cyber attack in comparison to the ground truth of the reliability of the AD algorithm established in the safety testing. This clearly demonstrated the effect of cyber-attacks regardless of the reliability of the algorithm. We were also able to assess, from the performance of the AD algorithm, that the algorithm is not optimised for the overtaking manoeuvre. In our research, we discovered several sensitive parameters that play a significant role in the safety outcome of the AV and the success rate of the cyber attack. Furthermore, we provided the results of our testing platform to the designer of the open-planner algorithm. Based on their feedback a process has been initiated to optimise the AD algorithm. All test scripts and software resources including our AV simulation configurations and research data used in the combined safety and security testing will be available for the research community on GitHub.

7.1 Future Work

Future work consists of diversifying the safety scenarios to include a more complex and broader range of scenarios. Cybersecurity testing will be evolved to develop black-box testing models. Furthermore, we will continue to develop methods for optimising testing to factor in real-world limitations such as resource usage and time.

ACKNOWLEDGMENTS

This research has received funding from the following grants: the European Union's Horizon 2020 Research and Innovation Programme, under grant agreements No. 856602 and No 883321 (CityScape), and the European Regional Development Fund, co-funded by the Estonian Ministry of Education and Research, under grant agreement No 2014-2020.4.01.20-0289.

REFERENCES

[1] Euro NCAP Working Group on Automated Driving. 2019. *Euro NCAP's First step to assess automated driving systems.* Technical Report. European New Car Assessment Programme.

[2] Adith Boloor, Karthik Garimella, Xin He, Christopher Gill, Yevgeniy Vorobeychik, and Xuan Zhang. 2020. Attacking vision-based perception in end-to-end autonomous driving models. *Journal of Systems Architecture* 110 (2020). https://doi.org/10.1016/j.sysarc.2020.101766 arXiv:1910.01907

[3] Yulong Cao, Chaowei Xiao, Benjamin Cyr, Yimeng Zhou, Won Park, Sara Rampazzi, Qi Alfred Chen, Kevin Fu, and Z. Morley Mao. 2019. Adversarial Sensor Attack on LiDAR-Based Perception in Autonomous Driving. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (London, United Kingdom) *(CCS '19)*. Association for Computing Machinery, New York, NY, USA, 2267–2281. https://doi.org/10.1145/3319535.3339815

[4] CARLA Simulation Project. 2022. *Combined Safety and Cybersecurity Testing Methodology for Autonomous Driving Algorithms.* Technical Report. CARLA.

[5] Hatem Darweesh, Eijiro Takeuchi, and Kazuya Takeda. 2021. OpenPlanner 2.0: The Portable Open Source Planner for Autonomous Driving Applications. In *2021 IEEE Intelligent Vehicles Symposium Workshops (IV Workshops)*. 313–318. https://doi.org/10.1109/IVWorkshops54471.2021.9669253

[6] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. 2017. CARLA: An Open Urban Driving Simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*. 1–16.

[7] Junyao Guo, Unmesh Kurup, and Mohak Shah. 2020. Is it Safe to Drive? An Overview of Factors, Metrics, and Datasets for Driveability Assessment in Autonomous Driving. *IEEE Transactions on Intelligent Transportation Systems* 21, 8 (2020), 3135–3151. https://doi.org/10.1109/TITS.2019.2926042

[8] R. Spencer Hallyburton, Yupei Liu, Yulong Cao, Z. Morley Mao, and Miroslav Pajic. 2022. Security Analysis of Camera-LiDAR Fusion Against Black-Box Attacks on Autonomous Vehicles. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 1903–1920. https://www.usenix.org/conference/usenixsecurity22/presentation/hallyburton

[9] Zhongyuan Hau, Kenneth T Co, Soteris Demetriou, and Emil C Lupu. 2021. Object Removal Attacks on LiDAR-based 3D Object Detectors. In *Third International Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*.

[10] ISO/TC 22/SC 32 Electrical and electronic components and general system aspects. 2018. *ISO 26262-1:2018 Road vehicles — Functional safety.* Technical Report. International Standards Organization.

[11] Shinpei Kato, Shota Tokunaga, Yuya Maruyama, Seiya Maeda, Manato Hirabayashi, Yuki Kitsukawa, Abraham Monrroy, Tomohito Ando, Yusuke Fujii, and Takuya Azumi. 2018. Autoware on board: Enabling autonomous vehicles with embedded systems. In *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 287–296.

[12] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and X. Zhang. 2018. Trojaning Attack on Neural Networks. In *NDSS*.

[13] Ehsan Malayjerdi, Raivo Sell, Mohsen Malayjerdi, Andres Udal, and Mauro Bellone. 2022. Practical path planning techniques in overtaking for autonomous shuttles. *Journal of Field Robotics* 39, 4 (2022), 410–425.

[14] Mohsen Malayjerdi, Vladimir Kuts, Raivo Sell, Tauno Otto, and Barış Cem Baykara. 2020. Virtual Simulations Environment Development for Autonomous Vehicles Interaction. In *ASME International Mechanical Engineering Congress and Exposition*. American Society of Mechanical Engineers.

[15] Mohsen Malayjerdi, Andrew Roberts, Olaf Maennel, and Ehsan Malayjerdi. 2022. *Combined Safety and Cybersecurity Testing Methodology for Autonomous Driving Algorithms.* https://github.com/momala/Safety_Cyber_Testing.git

[16] Raivo Sell, Ehsan Malayjerdi, Mohsen Malayjerdi, and Baris Cem Baykara. 2022. Safety Toolkit for Automated Vehicle Shuttle -Practical Implementation of Digital Twin. In *2022 International Conference on Connected Vehicle and Expo (ICCVE)*. 1–6. https://doi.org/10.1109/ICCVE52871.2022.9742881

[17] Jiachen Sun, Yulong Cao, Qi Alfred Chen, and Z. Morley Mao. 2020. Towards robust LiDAR-based perception in autonomous driving: General black-box adversarial sensor attack and countermeasures. *Proceedings of the 29th USENIX Security Symposium* (2020), 877–894. arXiv:2006.16974

[18] KWOK-LEUNG TSUI. 1992. AN OVERVIEW OF TAGUCHI METHOD AND NEWLY DEVELOPED STATISTICAL METHODS FOR ROBUST DESIGN. *IIE Transactions* 24, 5 (1992), 44–57. https://doi.org/10.1080/07408179208964244 arXiv:https://doi.org/10.1080/07408179208964244

[19] James Tu, Huichen Li, Xinchen Yan, Mengye Ren, Yun Chen, Ming Liang, Eilyan Bitar, Ersin Yumer, and Raquel Urtasun. 2021. Exploring Adversarial Robustness of Multi-sensor Perception Systems in Self Driving. In *5th Annual Conference on Robot Learning*. https://openreview.net/forum?id=m5k1XdK5nI2

[20] Kaichen Yang, Tzungyu Tsai, Honggang Yu, Max Panoff, Tsung-Yi Ho, and Yier Jin. 2021. Robust Roadside Physical Adversarial Attack Against Deep Learning in Lidar Perception Modules. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security* (Virtual Event, Hong Kong) *(ASIA CCS '21)*. Association for Computing Machinery, New York, NY, USA, 349–362. https://doi.org/10.1145/3433210.3453106

[21] Yi Zhu, Chenglin Miao, Tianhang Zheng, Foad Hajiaghajani, Lu Su, and Chunming Qiao. 2021. Can We Use Arbitrary Objects to Attack LiDAR Perception in Autonomous Driving? *Proceedings of the ACM Conference on Computer and Communications Security* (2021), 1945–1960. https://doi.org/10.1145/3460120.3485377

Appendix X

Paper X

A. Roberts, O. Maennel, and N. Snetkov. Cybersecurity test range for autonomous vehicle shuttles. 2021 IEEE European Symposium on Security and Privacy Workshops (EuroS PW), pages 239–248, 2021.

2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW) | 978-1-6654-1012-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/EuroSPW54576.2021.00031

Cybersecurity Test Range for Autonomous Vehicle Shuttles

Andrew Roberts, Olaf Maennel
School of IT, Department of Software Sciences
Tallinn University of Technology
Tallinn, Estonia
{*andrew.roberts,olaf.maennel*}*@taltech.ee*

Nikita Snetkov
Information Security Institute
Cybernetica AS
Tallinn, Estonia
nikita.snetkov@cyber.ee

Abstract—**Autonomous vehicle (AV) shuttles for public transportation are challenged by a lack of cybersecurity testing and platforms to conduct testing. Real-world, operational vehicles are expensive and operators are reluctant to test cybersecurity test cases that could impart damage to the systems. Furthermore, despite rigorous regression testing methods used within the broader automotive industry, vulnerabilities of automotive systems are still being found by creative edge and corner cases. To enable testing for edge and corner cases for AV shuttles, we propose the integration of cyber-physical test beds and ranges into the testing program of real-world AV shuttles. We evaluated the Massachusetts Institute of Technology (MIT) DuckieBot as the basis for comprehensive cybersecurity vulnerability testing and show how the results can be applied to the iseAuto, a real-world AV shuttle operating in Tallinn, Estonia. The MIT DuckieBot test bed is used to replicate the complexity and interactions of relevant systems of the iseAuto AV shuttle. The practical evaluation, involving cybersecurity edge and corner test cases demonstrated that cyber-physical test beds and ranges can support agile, repeatable cybersecurity testing that is low-cost and is conducted in a controlled and safe environment.**

Index Terms—**automotive cybersecurity, autonomous vehicle shuttles, cybersecurity testing, cybersecurity test bed, autonomous vehicle cyber range**

1. Introduction

Autonomous vehicle (AV) shuttles for public transportation are being piloted in European cities [1]. Cybersecurity of AV shuttles is of predominant importance for the safety of passengers and pedestrians in the traffic environment. Digitisation of vehicles and the transitioning to intelligent control by algorithms have exposed vulnerabilities to traditional cyber attacks such as ransomware, distributed denial of service, and new attack surfaces such as adversarial machine learning and sensor manipulation [2]–[5]. Recent examples [6], [7] of software failures of semi-autonomous vehicles resulting in fatalities of passengers have shown the lethal potentiality of cyber attacks. There are many challenges to securing AV shuttles against cyber attacks.

Cyber ranges are popular tools to experiment with edge and corner cybersecurity test cases and training for skills development and situational awareness of cybersecurity incident response. However, there is a lack of evaluation of cyber range technologies for AV cybersecurity and knowledge as to how cyber-physical systems can translate to support real-world, operational AV shuttles.

To address the challenges of AV cybersecurity, cybersecurity testing platforms for cyber-physical systems and methods for testing and training are required. In this paper we evaluate the Massachusetts Institute of Technology (MIT) Duckietown, low-cost, small-factor, cyber-physical AV test bed to support cybersecurity testing of a real-world AV Shuttle, operating in Tallinn, Estonia. The purpose is to understand how a cyber-physical test bed can be used for cybersecurity testing of AV shuttles and how this can transform cyber ranges and training for AV cybersecurity. Our main contributions are as follows:

1) We show the utility of a cyber-physical test bed for AV shuttles to support a real-world, operational AV shuttle.
2) We demonstrate, through a series of practical cybersecurity test scenarios, that a low-cost, cyber-physical test bed can be used to test the general cybersecurity of an AV shuttle and improve issues with the architecture and training for situational awareness of operators.
3) We outline recommendations how a cyber-physical test bed can be used to validate cybersecurity edge and corner cases.

2. Related Work

Cyber-physical test beds for AVs have featured in numerous studies. However, the related work is focused on the design of the test bed and there are few works that include considerations for cybersecurity testing and training.

Three studies are prominent in the related literature for their focus on designing low-cost cyber-physical test beds for automotive.

Axelsson *et al.* created a vehicle test bed for security evaluation of cyber physical system. The test bed was based on a small-factor mobile vehicle which was customised to support AUTOSAR, a software framework for automotive. The vehicle test bed, developed in 2014, demonstrated that a small-factor device could provide a solution to emulate the protocols and features of a full-factor real-life vehicle. The test bed was not autonomous and relied on remote control by human operator [29].

© 2021, Andrew Roberts. Under license to IEEE.

DOI 10.1109/EuroSPW54576.2021.00031
Authorized licensed use limited to: Tallinn University of Technology. Downloaded on June 13,2025 at 16:07:42 UTC from IEEE Xplore. Restrictions apply.

Tian developed a low-cost cyber-physical AV for research of neural networks. The research involved creating a code base for a line following car in a low-noise, controlled, test environment. The study developed the environment which could have applications for test bed and cyber range capability. However, as this was not the primary focus of the study, the translation of the cyber-physical AV for testing was not explored [30].

Bhadani *et al.* created a Cognitive and Autonomous Test (CAT) Vehicle test bed to evaluate AVs. The research problem highlighted in the study was the cost, time and risks of real-world testing and the problems translating test cases from simulators to real-world environments. The study designs and builds a hybrid virtual-physical test bed that incorporates the body physics of a real world vehicle with virtualised sensors and software platforms. ROS is used as the middleware platform. The evaluation of the platform was conducted through an educational program where students used extracted data from the CAT vehicle to improve object detection and tracking. The study was focused on the design of the vehicle and not cybersecurity, testing or training [32].

Zelle *et al.* and Santos & Schoop extended the design of a test bed for AVs to include a framework for cybersecurity testing of AVs. Both of these studies focused on test cases generated from either formal methods or system analysis [31], [33]. Zelle *et al.* built a security test platform for AVs using small-factor cyber-physical systems. The methods used in designing the platform comprised eliciting an attack model of cybersecurity attacks against autonomous vehicles. Based on this attack model the test bed was designed. The test bed is innovative, it includes most of the diverse range of sensors used for perception as well as in-vehicular networks and infotainment systems. The contribution is closest to this work. The main differentiation is that this study provided a practical assessment of the test bed and analysed testing and training methods that a cyber-physical AV test bed could support [31].

Santos & Schoop developed a framework for cybersecurity testing of AVs and evaluated its efficiency through investigation of the survivability of autonomous vehicles after a cyber attack to the vehicles sensors. Their framework consisted of developing test cases from formal methods and a tool to auto-generate test cases. Their practical evaluation involved the security testing of two sensors; camera and LiDAR. An open-source autonomous driving simulator, CARLA, was used as the experimental testing environment. The authors tool for automatic test case generation only supports CARLA. Their study acknowledges the limitations of this approach, the attack to the sensors was delivered by manual scripts and assumed the attackers could manipulate the sensors perfectly each time. The findings are limited to the CARLA environment and the simulation environment testing could not replicate a real-world physical attack or the operational driving domain of the vehicle [33].

3. AV Shuttle Cybersecurity Program

To select a low-cost, cyber-physical test bed to evaluate for AV shuttle cybersecurity, we begin by providing an overview of the Tallinn, Estonia, iseAuto, a real-world, operational AV shuttle.

AV shuttles are a type of AV used for public transportation in predominantly urban environments. AV shuttles can accommodate up to 15 passengers and are limited is speed to approximately 25 km/h. Table 1 lists a few of the different models of operational AV shuttles. There are thousands of AV shuttles currently operating around the world [1]. Figure 1 depicts a public transport AV shuttle.

TABLE 1. AV SHUTTLES FOR PUBLIC TRANSPORTATION [1]

AV Shuttle	Location	Environment
Navya Arma	Parc Olympique Lyonnais, France	Public Road
EasyMile EZ10	Airport Velizy-Villacoublay, Paris, France	Government Property
iseAuto	Tallinn, Estonia	Private Road
Baidu Apollo	Software Park Xiamen, China	Public Road
Local Motors Olli	Goodyear, Colmar-Berg, Luxembourg	Private Road

Figure 1. iseAuto Public transport AV Shuttle [8]

AV Shuttles use either open-source or proprietary software designed for AVs. The Robotic Operating System (ROS) is one of the key open-source systems. ROS is an open-source middleware that provides support for hardware abstraction, low-level device control, message-passing between processes, and package management. ROS is popularly used as it integrates with Autoware, a large open-source research and development community that provides a software platform for autonomous driving. The Autoware platform provides modules for self-driving, these include localisation, detection, prediction, planning and control [9]. These modules are essential for the vehicle to understand where it is located in the driving environment, map the route it must drive and detect the objects in the driving environment such as pedestrians. Furthermore, the control module is crucial for the vehicle to coordinate the conditions under which the control of the vehicle will be maintained and important decisions will be made, such as when control of the vehicle will be passed back to the human operator.

The AV shuttle architecture integrates this software ecosystem with advanced hardware technology and sensors: light detection and ranging (LiDAR), ultrasonic radar, camera and global navigation satellite system (GNSS). AV Shuttles mainly operate in Level 3 and 4 autonomy mode. SAE J3016 taxonomy and definitions of autonomy [10] classifies level 3 and level 4 as follows:

- *Level 3 (Conditional Driving Automation):* Driver does not have to monitor the system at all times, must always be in a position to resume control.

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on June 13,2025 at 16:07:42 UTC from IEEE Xplore. Restrictions apply.

System has longitudinal and lateral control in a specific case. System recognised the performance limits and request driver to resume control within a sufficient time margin [10].

- *Level 4 (High Driving Automation):* Driver is not required during the defined use case. The system can cope with all situations automatically in a defined use case [10].

AVs use teleoperation. Teleoperation is the remote monitoring and controlling of the AV by a human operator. In the real-world vehicle used in this study, the teleoperation is a software module of the ROS, enabling communication between the on-board computer and a remote teleoperation centre located in a building along the private road environment.

AV Shuttles are densely interconnected. The internal self-driving vehicle network consists of layers of communicating devices from the embedded components of the vehicle, including the electronic control units (ECUs) using the CAN Bus protocol, to the IP connected sensors. The vehicle communicates with smart road-sign-units (RSUs) and internet-connected devices, which is termed vehicle-to-everything (V2X), and with other vehicles, known as vehicle-to-vehicle (V2V).

The AV shuttles autonomous driving cognition and sensonics are tested in simulators and cyber-physical test beds. Popular simulators include; Apollo Baidu, Autoware lgsvl, CARLA and ROS Gazebo [11]. Simulators consist of a 3D generated driving environment, normally from the maps generated by LiDAR sensor. The simulated AV can take as input the same configurations used in the ROS software of the real-world vehicle and similar sensor software profiles. Cyber-physical test beds can be either small-factor replicas or hardware-in-the-loop test benches. Cyber-physical test beds allow the same features and functionalities of the simulated environment with the additional benefit of providing testing of physical interfaces and the dynamic of real-world physical conditions.

In 2015, researches demonstrated that the in-vehicle network, Controller Area Network (CAN) Bus, of a Jeep Cherokee could be exploited through malware and remote code injection, to stop the brakes of the vehicle [12]. This event precipitated the increase in focus on testing methods and test platforms for CAN Bus and connected vehicle technologies; communication interfaces and infotainment systems. This increase in research activity has lead to an increasing amount of vulnerabilities found in connected vehicles (Table 2).

There has been growing research in cybersecurity vulnerabilities of autonomous driving. These mainly focus on adversarial machine learning that aims to exploit weaknesses in the autonomous driving cognition, fuzzing of ROS and other middleware software, and network interfaces used for V2V and V2X communication (Table 3). Most of this research is conducted in simulators or on isolated systems and components and very few of the testing methods relate to real-world operational vehicles [2]–[5], [16], [17]. An exhaustive list of vulnerabilities of connected vehicles and AVs can be found here [18]

Despite commonly used regression testing methods and standards for cyber assurance testing of AVs, the vulnerabilities of AV systems continue grow.

TABLE 2. Examples of Cyber Attacks on Connected Vehicles

Vehicle	Cyber Threat
Tesla Model S	Spoof Passive keyless entry to take advantage of weak cryptography, lack of mutual authentication for challenge-response and lack of firmware protection [13].
	Malicious firmware with linux kernel exploitation for the ConnMan open-source internet connection manager allows WiFi of the Tesla to be exploited to allow remote connections [14].
Jeep Cherokee 2014	Malware on Infotainment system to allow remote root privileges and pivot into CAN Bus network [12].
KIA	Reverse-engineered Android OS Infotainment system. Found vulnerabilities to allow remote root privileges [15].

TABLE 3. Examples of Cyber Attacks on AVs

Attack Surface	Cyber Threat
Autonomous Cognition	Tamper with RSU Stop Sign to manipulate autonomous cognition [19].
	Tamper with lane markings to manipulate lane-keeping system(LKAS) [20].
	Spoofed images in driving environment to manipulate object-detection [21].
Sensors	Jam LiDAR point cloud sensor with laser [22].
	Tamper with sensor data to manipulate navigation path [23].
System	Spoofing of ROS Master and interception of ROS messages [24]
	Malware in firmware update [25]
	Fuzzing of AV middleware [26].
Network	Intercept and spoof RSU messages [2]

Public transportation AV shuttles undergo limited testing for cybersecurity, this is due to many reasons. Firstly, cybersecurity testing on real-world proving grounds with operational vehicles is expensive and time-consuming, requiring extensive labour effort in the setup, execution and safety monitoring of the tests [27]. Secondly, there is a reluctance to test cybersecurity test cases that could damage the vehicle. This is mainly due to the cost and time involved in rebuilding and re-configuring vehicular systems and components. Thirdly, the AV shuttle architecture is a distributed systems architecture and due to lack of testing there is a gap in understanding how cyber attacks cause cascading affects and how, for instance, malware could propagate throughout the system. Fourthly, there is a lack of investigation of novel testing methods and techniques for cybersecurity. These include software simulators and cyber-physical test beds commonly used for testing autonomous driving cognition. Lastly, there is lack of training of teleoperation, remote control vehicle operators for situational awareness for cybersecurity. As AV shuttles rely on teleoperation operators to override the autonomous cognition in emergency situations and make manual driving decisions, their awareness as to how cyber attacks can impact situational awareness is critical for safe driving operation.

Flexible testing environments that allow agile testing of edge and corner cyberseecurity test scenarios would help assist in identifying vulnerabilities of the AV system architecture. Whilst simulators and small-factor cyber-

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on June 13,2025 at 16:07:42 UTC from IEEE Xplore. Restrictions apply.

physical test beds are used for testing and improving autonomous driving algorithms there has been limited practical exploration of these test beds for cybersecurity testing.

Test beds such as the MIT DuckieTown, provide a low-cost, small-factor environment accessible to autonomous self-driving vehicle developers and quality assurance testers [28]. These environments, which utilise the same software and network interfaces as Autonomous Vehicle (AV) Shuttles have the potential to be used for cybersecurity testing and research.

4. Cybersecurity Test Beds for AV Shuttles

4.1. Test Bed Requirements

Key factors which influence the design and usage of test beds to support the operational AV shuttle include cost, complexity and fidelity of the test bed to the operational system.

4.1.1. Cost. To support agile testing and cybersecurity test cases that impart physical damage on the AV, the cost of the test bed needs to be as limited as possible. The low-cost requirement has two intended beneficiaries. Firstly, a low-cost agile test bed can be given to students and researchers in innovative testing formats such as crowd sourcing. This enables a wider scope of testing for minimal cost. Secondly, AV shuttle programs for public transportation do not have exhaustive resources for testing in comparison to the major original equipment manufacturers. Therefore, low cost test beds are required to test edge and corner cases and prioritise test cases for testing on the real-world vehicle.

4.1.2. Complexity. AV shuttles are a complex distributed system architecture, it is essential that the test bed support observation of distributed system interaction whilst limiting the complexity to allow rebuilding of damaged systems. For example, allowing a clean rebuilding of a software or hardware system infected by malware. This agility will allow repeatable testing of cybersecurity test cases and enable dynamic testing such as crowdsourced vulnerability analysis and training such as capture-the-flag style learning activities.

4.1.3. Fidelity to Operational Vehicle. To evaluate cybersecurity and situational awareness there needs to be a level of abstraction of the operational vehicle architecture. An evaluation of the real-world AV shuttle considered the AI & Drive systems, sensonics and the network connectivity with the teleoperation as key features of the autonomous driving architecture to emulate in a test bed.

4.2. Test Bed Analysis

A comparison of test beds used for autonomous driving and cybersecurity research found the cyber-physical test bed to be an optimal platform for evaluation (Table 4). Advantages of the cyber-physical system are the low cost and agile, modular architecture which can allow sensors and systems to be added or removed. Due to the lack of evaluation of cyber-physical test beds to support cybersecurity testing their fidelity to real-world, operational system is yet to be determined, and will be explored in this study.

TABLE 4. Comparison of Test Bed Architectures to support Cybersecurity

Testing Considerations	Simulation	Cyber-Physical	Real-World Proving Ground
Cost	Low	Low	Medium
Complexity	Low	Medium	High
Fidelity	Low	Not evaluated	High

Table 5 presents an evaluation of the test bed architectures to support testing for the cyber attacks listed in Table 2 and Table 3.

TABLE 5. Comparison of Test Bed Architectures to support AV Shuttle cybersecurity Test Cases

Cyber Threat Test Cases	Simulation	Cyber-Physical	Real-World Proving Ground
Hardware and Compute	Yes	Yes	Yes
Connected Vehicle	Yes	Yes	Yes
Sensor and Perception	Yes	Yes	Yes
Physical Access	No	Yes	Yes
Damage Incurring	No	Yes	Yes
Environmental Perturbations	No	Yes	Yes

4.3. AV Shuttle Cyber Range for Cybersecurity

The MIT CSAIL Duckietown is a small-factor test bed used for evaluating autonomous driving software modules, algorithms and education. Duckietown consists of a driving environment (Figure 2) and an AV, called, DuckieBot (Figure 3).The cost of the components to build the MIT Duckietown test bed is approximately €400.

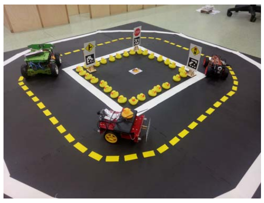


Figure 2. DuckieTown - Cyber-Physical Test Bed

The DuckieBot uses a 5 mega pixel Raspberry Pi camera for sensing. The hardware for the AI and Drive Algorithm is built on Raspberry Pi Model 3B hardware. The software platform is built upon Docker utilising ROS Kinetic. A 32GB SD card is used for local on-board storage and a 100Gb USB drive can be inserted in the Raspberry Pi to allow more storage for logging. A 5 volt, 10400 mAh, battery is used to power the DuckieBot. Actuation is performed by the motor driver which connects to

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on June 13,2025 at 16:07:42 UTC from IEEE Xplore. Restrictions apply.

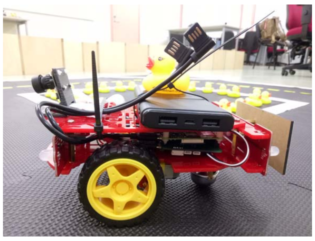


Figure 3. Duckiebot - AV Cyber-Physical Device

servo motors. The DuckieBot is calibrated to operate in the DuckieTown driving environment. This consists of a floor layer with road markings, conventional to the standard markings of real-world traffic.

Table 6 represents detailed analysis of the DuckieBot with the iseAuto AV shuttle, operating in Tallinn, Estonia.

TABLE 6. Feature Comparison of Test Bed and iseAuto AV Shuttle

DuckieBot	iseAuto AV Shuttle
ROS Kinetic Kame	ROS Kinetic Kame
Linux Network Interfaces and 4G Cellular Network	Linux Network Interfaces and 4G/5G Cellular Connectivity (*V2X is yet to be added as a feature)
Camera Sensing	Camera, LiDAR, Ultrasonic Radar, GNSS
Actuation, motor driver controls servo motors	Actuation, Drive Controller controls CAR ECU
On-board Control PC (ARM processor)	On-board Control PC (ARM processor) different hardware specifications
Teleoperation - Mission Control System	Teleoperation - Mission Control System

The DuckieBot is an optimal test bed for experimentation as it uses the containerised architecture of Docker. This allows software packages for sensors, hardware and AD to be centralised in a configurable system. This enables packages to be added or removed depending on the test case and for new sensors and hardware to be added easily. The other major advantage is that the DuckieBot is an actively supported open-source project and new AD algorithms are published regularly. This helps to ensure that test cases are tested against the newest available AD algorithms.

5. Cybersecurity Test Scenarios for AV Shuttles

5.1. Test Scenario Generation Process

Some example use-cases are used to evaluate the usefulness of the cyber-physical range. To generate the cyber test scenarios we asked experts in AV cybersecurity from vehicle manufacturers and system designers to detail edge and corner cybersecurity test cases that they would want a AV cybersecurity test bed to support. The experts represented organisations that develop autonomous robots for logistics, autonomous driving assistance systems and AV shuttle operators. Table 7 lists illustrates our chosen demonstration use-cases.

TABLE 7. Security Test Case Scenarios

	Test Case
Scenario 1 (S01)	An external threat actor spoofs the road markings to manipulate the driving logic to veer the vehicle off the road.
Scenario 2 (S02)	An external threat actor tampers with the road markings to manipulate the drive logic to veer the vehicle off the road.
Scenario 3 (S03)	An external threat actor tampers with the camera sensor using a laser pointer to blind or shield it's perception to manipulate the drive logic to veer the vehicle off the road.
Scenario 4 (S04)	An external threat actor spoofs the RSU to manipulate the drive logic to veer the vehicle off the road.
Scenario 5 (S05)	An innocent maintenance engineer executes a malicious cryptocurrency or ransomware malware hiding as a firmware update for a vehicle system.
Scenario 6 (S06)	An external threat actor eavesdrops on the ROS vehicular messaging system for information gathering
Scenario 7 (S07)	An external threat actor attacker conducts a denial-of-service of the teleoperation communication link with the AV.
Scenario 8 (S08)	An external threat actor uses a smoke gun to perturb the camera sensor vision and alter the driving course of the vehicle

5.2. Scenarios

In this section, we detail a few of the test scenarios. The aim of the test scenarios is to understand the utility of cybersecurity testing in an AV cyber-physical testbed environment to the real-world AV shuttle. The verification of the test results does not focus on a binary, yes/no conclusion, rather, a deeper analysis of whether the behaviour of the AV system observed during the cybersecurity testing can be used to identify vulnerabilities of the real-world AV shuttle architecture to cyber attacks. All of the scenarios can be viewed at the YouTube channel that was created to demonstrate the AV cyber range: https://tinyurl.com/2xxvvkzd [34]

5.2.1. S01 - Projected Road Markings. Problem: The projector attack consists of an attacker crafting a spoof image to be projected onto the traffic environment. The aim for the attacker is to fool the autonomous drive cognition to accept the spoofed image as genuine and alter the driving behaviour. An example would be a project of a lane marking on the road to alter the course of the AV. The projection attack experiments as detailed in Nassi *et al.* [21], used trial-and-error as a method of testing. The attack was trialled on real-world vehicles in a private campus environment. The testing environment was tightly controlled for safety reasons and the setup of the test took considerable time and effort. In DuckieTown, this attack can be tested and repeated using as many diverse methods as possible. The small, cyber-physical testing environment allowed for agility and repeatability and enabled replication of a cyber threat identified in a paper to test the validity of the results to our Av shuttle.

Whilst a spoofing attack using projection is a novel and interesting method to manipulate an autonomous vehicle it is unlikely or has low probability of success. The projection must contend with natural light, which means

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on June 13,2025 at 16:07:42 UTC from IEEE Xplore. Restrictions apply.

the attack must be undertaken at night. DuckieTown can be used for situational awareness for projections and spoofed images in the training of teleoperation operators. They must understand that these attacks can occur and have the ability to confuse the human operator into thinking the autonomous cognition has failed to detect a lane marking.

Scenario: An external threat actor spoofs the road markings to manipulate the drive logic to veer the vehicle off the road.

Attack Sequence:

1) Attacker observes the autonomous self-driving vehicle to understand how the autonomous drive cognition makes decisions.
2) Attacker crafts a spoofed image of a lane marking for projection on the driving environment.
3) Attacker positions a drone with a projector attached to it, in proximity to the vehicle and uses a remote control to initiate the projection attack.

Results: The spoofed projection attacks were unable to alter the driving actions of the DuckieBot. Figure 4 shows the faint image of the phantom spoofed yellow line which is barely visible due to the bright profile of the driving environment. Figure 5 visibly shows the spoofed line marking, due to a larger spoofed image being projected by the attacker. The figure 5 image, from the DuckieBot camera shows that the autonomous drive cognition is detecting the edges and texture of the yellow lines and white boundaries but is not detecting the spoofed projection image. This is due to the lack of edges, texture and geometry of the spoofed projection image.

Multiple diverse attack methods were trialled, the spoofed projection images were left projecting on the road surface for 10 minutes, the size of the images were increased, the definition of the images increased, projection on different sections of the floor and different environmental light. The DuckieBot was resilient to the projected road markings attack and the autonomous drive cognition was not manipulated.

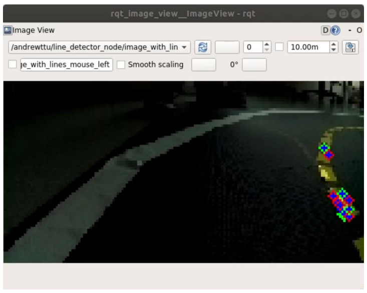

rqt_image_view__ImageView - rqt
Image View
/andrewttu/line_detector_node/image_with_lin
0
10.00m
je_with_lines_mouse_left
Smooth scaling
0°

Figure 4. Scenario 1 - Projected Lane Marking

5.2.2. S02 - Tampered Lane Markings. Problem: Although this threat seems simplistic in the experimental test bed environment, the implications for a real-world operational vehicle are stark. An attacker can use a 3D printer to print a tampered road patch and place it on the road markings of a highway. If this test had occurred on an

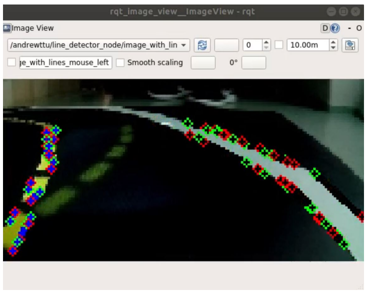

rqt_image_view__ImageView - rqt
Image View
/andrewttu/line_detector_node/image_with_lin
0
10.00m
je_with_lines_mouse_left
Smooth scaling
0°

Figure 5. Scenario 1 - Projected Lane Marking

autonomous vehicle travelling at 40 mph it would have resulted in physical damage to the AV. This attack shows the usefulness of the cyber-physical AV. The cyber-physical AV enabled this attack to be experimented repeatably and we were able to try different lane marking manipulations. This is an improvement on the methods used by Sato *et al.* [20] where they used a simulation for testing and this simulation environment wasn't able to replicate the role of the teleoperation or camera sensing. Through testing this attack in the DuckieTown, we can see that the teleoperation operator must maintain situational awareness of the road environment if there have been any manipulations by a threat actor or environmental damage. In translating this scenario to the real-world AV we were able to detect that the operational vehicle would be susceptible to this same attack. From this experiment, a greater examination of the sensing and detection algorithms of the real-world vehicle was conducted and updates to the multi-sensor fusion were made to mitigate the risk of this attack.

Scenario: An external threat actor tampers with the road markings to manipulate the drive logic to veer the vehicle off the road

Attack Sequence:

1) Attacker observes the autonomous self-driving vehicle to understand how the autonomous driving cognition makes decisions.
2) Attacker, using the understanding of the drive control algorithm, perturbs the road markings in the DuckieTown environment.

Results: Perturbation of a road marking can manipulate the drive algorithm to cause the autonomous self-driving vehicle to veer off the intended path of travel.

In the first experiment the attacker tampered with the yellow lane markers to manipulate the autonomous self-driving vehicle to drive off the road. The curve road part was changed to a straight trajectory and the angle of the lane borders (white lines) were reduced to lessen the width of the road. As Figure 7 demonstrates, the change to the road markings is demonstrable in the DuckieBot camera sensor footage, from the expected road markings exhibited in Figure 6. The first experiment was successful in manipulating the autonomous drive cognition of the DuckieBot, however, the DuckieBot autonomy is programmed to firstly respect the lane boundaries. The DuckieBot followed the tampered yellow line until it detected the lane boundary and then adjusted it's travel path to the correct route.

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on June 13,2025 at 16:07:42 UTC from IEEE Xplore. Restrictions apply.

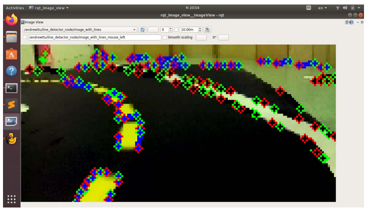


Figure 6. Normal Lane Markings

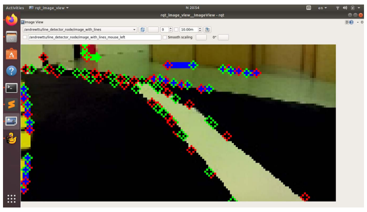


Figure 7. Manipulated Lane Marking

In the second and third the attacker extended the yellow lane markings further into the lane boundaries. The DuckieBot still respected the boundaries and corrected the path of travel.

In the fourth and fifth experiment the attacker removed the lane boundaries and extended the yellow lane markings, as shown in Figure 8 . The attack was successful and the DuckieBot veered off the DuckieTown environment and was unable to recover.

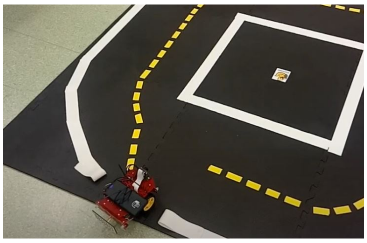


Figure 8. Extended Manipulated Lane Marking

5.2.3. S05 - Firmware update compromise. Problem: Malware in a distributed system provides interesting observations, an autonomous vehicle could lose access to a secure network and connect to a more vulnerable network which would allow malware to propagate more extensively. In testing the findings of Weiss *et al.* [25], we were able, in DuckieTown, use real malware and observe how it propagates in an AV system. We were able to clean the system and repeat the attack to observe any differences in behaviour. In translating the results of the malware attack to the real-world AV shuttle, the AV engineers were unaware of the risks posed in connecting wireless networks in the transit environment. There are many applications for this range scenario for training. Firstly, this scenario would be useful to test incident response to malware in AVs. Secondly, it would be beneficial for the engineers to understand the risks posed by a lack of validation of firmware updates and how malware can spread within a distributed system.

Scenario: An innocent maintenance engineer executes a malicious cryptocurrency or ransomware malware hiding as a firmware update for a vehicle system created by an angry mechanic/insider.

Attack Sequence:

1) Angry Mechanic uploads malware from dark web or publically available repository.
2) Malware script is packaged as a bash script that is labelled ”update”.
3) Maintenance engineer initiates ”update” script with intention of update AV shuttle firmware.

Results: The ”update” firmware was executed by the innocent maintenance engineer. The update firmware contained a bash script which executed a cryptomining program. Once infected on a host computer the malware installs several libraries and processes for it’s operation and then tries to install zmap (net-work scanner) and ssh pass (utility for establishing SSH connections). It uses zmap, in an infinite loop,to discover then network and find embedded devices with port 22 (SSH) open. If these are found, it connects to the device using ssh with default passwords. It then changes the configuration settings of the device to allow a connection to a command and control node used for cryptomining. On the DuckieBot, the malware installed it’s libraries and zmap and ssh pass and began a zmap scan of the network. The DuckieBot was on a private 4G network that also had another AV connected. As these devices do not use default passwords it was unable to establish a connection to them. The zmap scans only marginally impacted the performance of the network of the DuckieBot. The zmap scan was sending 50,000 packets to the target device, but these are only looking for open port 22. An interesting event happened during the experiment. The 4G cellular private network lost connection during the malware execution and the DuckieBot switched over to an open wireless network connection (controlled by us). The zmap process then started to scan the network for open embedded devices. The malware attack was attempted again and this time the wireless campus network was removed. The malware behaved in the same manner and was unsuccessful in brute forcing the DuckieBot.

5.2.4. S06 - Eavesdropping of AV Shuttle operations. Problem: ROS is highly insecure. The version that the DuckieBot is running is the same as the vehicles used by real-world AV Shuttles. There is no authentication and secure communication of the ROS Master. The ROS Master also uses HTTP so it is vulnerable to a number of other malicious web application attacks. The AV testing environment enabled us to test on a real-time system to understand dynamically the information that can be gathered from reading ROS messages and the possibilities of how this information can be used to develop an attack on the vehicle platform. In translation of this to our real-world AV, the mitigating action is to filter the ROS port with a firewall rule. However, if the attacker gains access to the internal network of the AV system there is little possibility to prevent this attack other than to upgrade

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on June 13,2025 at 16:07:42 UTC from IEEE Xplore. Restrictions apply.

to the latest version of ROS, ROS2, which is still under development.

Scenario: An external threat actor eavesdrops on the ROS vehicular messaging system for information gathering.

Attack Sequence: For this attack, the attacker needs to be on the same network as the vehicle.

1) Attacker scans the network and identifies the vehicle
2) Attacker eavesdrops on the ROS communication by spoofing the ROS Master.

Result: The attacker was able to spoof the ROS Master easily and read the ROS messages which are used for AV operations. The attacker was able to generate a ROS graph that showed all of the communication ROS messages (picture not shown/included for anonymity reasons). From this, the attacker could develop a diverse range of attacks such as injection of ROS commands to manipulate a ROS node and replay attacks.

5.2.5. S07 - DDoS Teleoperation Network. Problem:

The DDoS attack is an important scenario to replicate in a cyber range due to the loss of control of the teleoperation operator to safely stop the vehicle. This scenario was interesting for the real-world AV shuttle teleoperation staff. When the DDoS attack was conducted the teleoperation console froze and only when the network was re-established did they see that the AV had crashed. This scenario is important for situational awareness training.

Scenario: An external threat actor conducts a denial of service of the short-range wireless network of the autonomous self-driving vehicle.

Attack Sequence:

1) Attacker scans wireless and cellular networks of the vehicle using scanning software such as nmap or airmagnet.
2) Attacker finds the WiFi access point connecting to the human operator console and autonomous self-driving vehicle.
3) Attacker De-authenticates the devices connected to the WiFi access point.

Results: A scan of all wireless networks was conducted on the attackers PC. The attacker used a wireless scanning device that can be considered a malicious access point that acts as a man-in-the-middle between the wireless network and the client device. It can scan, capture traffic and execute a number of attacks such as capturing passwords of insecure network protocols.

The deauthentication attack was attempted twice. Both attempts were successful. Figure 9 shows the teleoperation console after it loses access to the network connection with the DuckieBot and the DuckieBot accelerates off the road. Figure 10 shows the DuckieBot impacting the wall when it loses connectivity. The DuckieBot continues to accelerate on hitting the wall.

5.2.6. S08 - Smoke machine sensor perturbation.

Problem: The expert from the autonomous robotics for logistics organisation requested this test case as they wanted to see environmental impacts on the cyber-physical system and how they can relate to their real-world autonomous

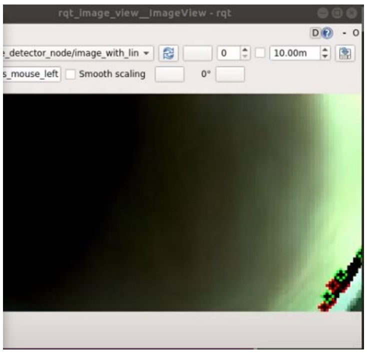

rqt_image_view__ImageView - rqt
_detector_node/image_with_lin
0
10.00m
s_mouse_left
Smooth scaling
0°

Figure 9. Camera Sensor Vision - DDoS Attack Crash

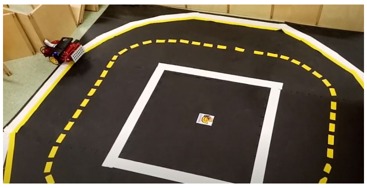


Figure 10. DuckieBot crashed after DDoS

systems. The test case demonstrated the utility of cyber-physical AV test bed in being able to simulate diverse environment conditions. Based on the results of the test case it may be possible to include safety testing in the scope of the test bed.

Scenario: An external threat actor uses a smoke machine to perturb the camera sensor vision and alter the driving course of the vehicle.

Attack Sequence:

1) A 400w smoke machine is placed next to the environment. The smoke machine is filled with special liquid and then activated using the command controller. Smoke envelops the driving environment.

Results: The experiments were conducted under three different lighting conditions: controlled lights, natural light, controlled dark lighting. In all lighting conditions the smoke was able to perturb the camera sensor to alter the driving path of the DuckieBot to crash out of the road environment.

The initial experimental tests, which were unsuccessful in altering the DuckieBot path, showed that the most important variables were the denseness of the smoke and the ability of the smoke to linger in the air to envelop the camera. The first three smoke experimental tests demonstrated the autonomous driving cognition being lost due to the smoke hazard, however, as the smoke stream was momentary, the detection of the lane markings were recovered in time to navigate accurately. Figure 11 shows the lane detection functioning, and Figure12 shows the smoke perturbing the lane detection of the lane markings.

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on June 13,2025 at 16:07:42 UTC from IEEE Xplore. Restrictions apply.

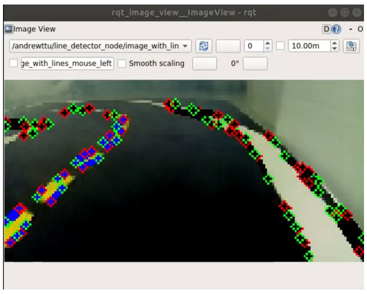

rqt_image_view__ImageView - rqt
Image View
/andrewttu/line_detector_node/image_with_lin
0
10.00m
ge_with_lines_mouse_left
Smooth scaling
0°

Figure 11. Lane Detection

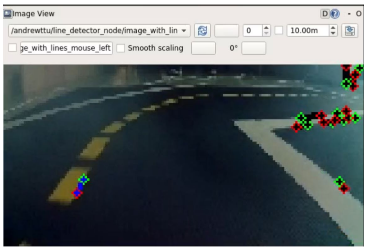


Figure 12. Smoke Perturbed Lane Detection

6. Discussion

The MIT DuckieTown cyber-physical AV shuttle test bed demonstrated it's use in validating the viability of proof-of-concept attacks such as that of the projector attack and the spoofed lane keeping assistance. The test bed enabled agility and repeatably of testing which facilitated greater understanding of the complexity of implementation of cyber attacks on AVs as well as the challenges for situational awareness for AV operators. A clear representation is the projector attack which demonstrated that it was very difficult for the adversary to accomplish due to the lighting, projection and camera angle requirements. The WiFi test case provided insights into possibilities for interoperability and human operator research. The vulnerabilities of the network interface, exploited in the cybersecurity test case, impacted the vehicle behaviour and human control.

The results of the testing were provided to the iseAuto, real-world AV Shuttle program. Based on the results analysis, the AV shuttle operator identified a number of vulnerabilities in the AV shuttle architecture. This resulted in the updating of the network package to stop the vehicle in the event of network unavailability or outage. Furthermore, the results helped to educate the teleoperation AV shuttle operators about some of the scenarios they could encounter from an adversarial actor in the driving environment and based on this it assisted in initiating a discussion on what decisions the operator would make when faced with a scenario such as the projection attack or environmental perturbations.

The feedback from the iseAuto concluded that the cyber-physical test bed offered a platform for which they could test corner and edge cases that would be out-of-scope of the real-world vehicle due to cost and risk impacts. It helped the iseAuto AV shuttle program in understanding how their AV Shuttle could be impacted by cyber attacks and with prioritising which attacks were most likely and require further testing on the real AV shuttle.

7. Conclusion

We demonstrated that cyber-physical test beds for AVs could be used to test edge and corner cybersecurity test cases. The results of these tests had applicability for the real-world, operational AV shuttle. Firstly, they identified areas in the AV architecture where a cyber threat actor could exploit vulnerabilities. Secondly, they identified areas where training could be conducted to improve situational awareness. We also demonstrated that the cyber-physical AV test bed offers advantages such as agile, repeatable testing which can be conducted safely, efficiently and at low-cost. As AV shuttles are challenged from a lack of cybersecurity testing, the cyber-physical test bed provides platform that can be used to improve AV cybersecurity.

8. Future Work

Future work concerns extending the use of the AV cyber-physical test bed to support innovative training and testing methods. The AV cyber-physical test bed will be transformed to integrate with a federated cyber range. The aim will be to extend the availability of the cyber-physical AV cyber range to a wider audience and evaluate its performance in a large cyber range environment. The integrated cyber range will then be tested using crowdsourced security testing methods. As AV shuttles are developed in open-source communities such as the Autoware Foundation, the crowdsourced security testing offers a method that can be leveraged to include AV engineers in the process of cybersecurity testing. Evaluation of the crowdsourced method to produce improved testing outcomes, training and awareness of cybersecurity will produce valuable insights for AV shuttle cybersecurity.

Acknowledgements

This work was partially supported by the European Commission through the H2020 project Finest Twins (grant No. 856602) and the ECHO project (grant agreement no 830943).

References

[1] Calin Iclodean, Nicolae Cordos, and Bogdan Ovidiu Varga. 2020. Autonomous Shuttle Bus for Public Transportation: A Review.Energies13, 11 (2020). https://doi.org/10.3390/en13112917

[2] Jonathan Petit and Steven Shladover. Potential cyberattacks on automated vehicles.Intelligent Transportation Systems, IEEE Transactions on, PP:1–11, 09 2014

[3] S. Parkinson, P. Ward, K. Wilson, and J. Miller. Cyber threats facing autonomous and connected vehicles: Future challenges.IEEE Transactions on Intelligent Transportation Systems, 18(11):2898–2915, Nov 2017.

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on June 13,2025 at 16:07:42 UTC from IEEE Xplore. Restrictions apply.

[4] Stephen Checkoway,Damon McCoy,Brian Kantor,Danny Anderson,Hovav Shacham,Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, and Tadayoshi Kohno.Comprehensive experimental analyses of automotive attack surfaces. In Proceedings of the 20th USENIX Conference on Security, SEC'11, page 6, USA, 2011. USENIX Association.

[5] V. L. L. Thing and J. Wu. Autonomous vehicle security: A taxonomy of attacks and defences. In2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (Smart Data), pages 164–170,Dec 2016.

[6] A.Davies (2019, May.).Tesla's Latest Autopilot Death Looks Just Like a Prior Crash, WIRED Magazine.Accessed 20 March,2020.[Online].Available:https://www.wired.com/story/teslas-latest-autopilot-death-looks-like-prior-crash/.

[7] A.Marshall (2018, Mar.). Uber's Self-Driving Car Just Killed Somebody. Now What?,WIRED Magazine. Accessed 20 May, 2020. [Online]. Available:https://www.wired.com/story/uber-self-driving-car-crash-arizona-pedestrian/

[8] Aripaev, *TalTech is developing a AV together with U.S university*, 2019. Accessed on: 5 May 2021. [Online]. Available:https://www.toostusuudised.ee/uudised/2019/04/10

[9] Autoware foundation, *The Autoware Foundation.* 2021. Accessed on: 10 May 2021. [Online]. Available: https://www.autoware.org/

[10] SAE International. SAE J3016 - Levels of Driving Automation. Accessed on 10 May 2021. [Online]. Available: https://www.sae.org/standards/content/j3016

[11] G. Rong et al., "LGSVL Simulator: A High Fidelity Simulator for Autonomous Driving," 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), 2020, pp. 1-6, doi: 10.1109/ITSC45102.2020.9294422.

[12] C.Miller & C.Valasek (2015, Aug.). Remote Exploitation of an Unaltered Passenger Vehicle. Accessed 20 June, 2020. [Online]. Available: http://illmatics.com/Remote%20Car%20Hacking.pdf

[13] Lennert Wouters, Eduard Marin, Tomer Ashur, Benedikt Gierlichs, and Bart Preneel. 2019. Fast, Furious and Insecure: Passive Keyless Entry and Start Systems in Modern Super cars.IACR Transactions on Cryptographic Hardware and Embedded Systems2019, Issue 3 (2019), 66–85. https://doi.org/10.13154/tches.v2019.i3.66-85

[14] Weinmann, R.P & Schmotzle, B, TBONE, 28 April 2021. Accessed on: 10 May 2021. [Online]. Available: https://kunnamon.io/tbone/

[15] Costantino, G & Matteucci, I.*KOFFEE - Kia OFFensivE Exploit*, 2020. Accessed on: 10 May 2021. [Online]. Available: https://www-old.iit.cnr.it/node/59491

[16] Sghiri Meryem and Tomader Mazri. Security study and challenges of connected autonomous vehicles. In Proceedings of the 4th International Conference on Smart City Applications, SCA '19, New York, NY, USA, 2019. Association for Computing Machinery.

[17] K. Ren, Q. Wang, C. Wang, Z. Qin, and X. Lin. The security of autonomous driving:Threats, defenses, and future directions.Proceedings of the IEEE, 108(2):357–372,2020.

[18] Shah Khalid Khan, Nirajan Shiwakoti, Peter Stasinopoulos, and Yilun Chen. 2020.Cyber-attacks in the next-generation cars, mitigation techniques, anticipated readiness and future directions.Accident Analysis Prevention148 (2020), 105837. https://doi.org/10.1016/j.aap.2020.105837

[19] Kevin Eykholt,Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City,UT,USA,June18-22,2018,pages1625–1634. IEEE Computer Society, 2018

[20] Takami Sato, Junjie Shen, Ningfei Wang, Yunhan Jia, Xue Lin, and Qi Alfred Chen.Security of deep learning based lane keeping system under physical-world adversarial attack, 03 2020.

[21] Ben Nassi, Yisroel Mirsky, Dudi Nassi, Raz Ben-Netanel, Oleg Drokin, and Yuval Elovici. 2020. 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(Virtual Event,USA)(CCS '20). Association for Computing Machinery, New York, NY, USA,293–308. https://doi.org/10.1145/3372297.3423359

[22] Yulong Cao, Chaowei Xiao, Benjamin Cyr, Yimeng Zhou, Won Park, Sara Rampazzi,Qi Alfred Chen, Kevin Fu, and Z. Morley Mao. Adversarial sensor attack on lidar-based perception in autonomous driving. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS '19, page 2267–2281,New York, NY, USA, 2019. Association for Computing Machinery.

[23] Drew Davidson, Hao Wu, Rob Jellinek, Vikas Singh, and Thomas Ristenpart. Control-ling uavs with sensor input spoofing attacks. In10th USENIX Workshop on Offensive Technologies (WOOT 16), Austin, TX, August 2016. USENIX Association.

[24] Se-Yeon Jeong, I-Ju Choi, Yeong-Jin Kim, Yong-Min Shin, Jeong-Hun Han, Goo-HongJung, and Kyoung-Gon Kim. A study on ros vulnerabilities and countermeasure. In Proceedings of the Companion of the 2017 ACM/IEEE International Conference on Human Robot Interaction, HRI'17, page147–148, NewYork, NY, USA, 2017. Association for Computing Machinery.

[25] Nils Weiss, Markus Schrötter, and Rudolf Hackenberg. On threat analysis and risk estimation of automotive ransomware. In ACM Computer Science in Cars Symposium,CSCS '19, New York, NY, USA, 2019. Association for Computing Machinery.

[26] Jia Cheng Han and Zhi Quan Zhou. 2020. Metamorphic Fuzz Testing of Autonomous Vehicles. In Proceedings of the IEEE/ACM 42nd International Confer-ence on Software Engineering Workshops(Seoul, Republic of Korea)(ICSEW'20).Association for Computing Machinery, New York, NY, USA, 380–385. https://doi.org/10.1145/3387940.3392252

[27] Vinzenz, N. and Oka, D., "Integrating Fuzz Testing into the Cybersecurity Validation Strategy," SAE Technical Paper 2021-01-0139, 2021, https://doi.org/10.4271/2021-01-0139.

[28] Jacopo Tani, Liam Paull, Maria T. Zuber, Daniela Rus, Jonathan How, John Leonard,and Andrea Censi. Duckietown: An innovative way to teach autonomy. In Dimitris Alimisis, Michele Moro, and Emanuele Menegatti, editors,Educational Robotics in the Makers Era, pages 104–121, Cham, 2017. Springer International Publishing.

[29] Axelsson, A. Kobetski, Z. Ni, S. Zhang, and E. Johansson. Moped: A mobile open platform for experimental design of cyber-physical systems. In 2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications, pages 423–430, 2014.

[30] D.Tian(2019,April.).Deep Learning, Self Driving Robotic Car on a Shoestring Budget.Accessed 20 March, 2020. [Online]. Available:https://towardsdatascience.com/deeppicar-part-1-102e03c83f2c

[31] D.Zelle, R.Rieke, C.Plappert, C.Krauß, D.Levshun, and A.Chechulin. Sepad security evaluation platform for autonomous driving. In 2020 28th Euromicro International Conferenceon Parallel, Distributed and Network Based Processing (PDP),pages413–420, 2020.

[32] Rahul Bhadani, Jonathan Sprinkle, and Matthew Bunting. The CAT Vehicle Test bed:A Simulator with Hardware in the Loop for Autonomous Vehicle Applications. In Proceedings 2nd International Workshop on Safe Control of Autonomous Vehicles(SCAV94 2018), Porto, Portugal, Electronic Proceedings in Theoretical Computer Science, volume 269, 04/2018 2018.

[33] Eduardo dos Santos and Dominik Schoop. 2018. Towards a Simulation-based Framework for the Security Testing of Autonomous Vehicles. In 6th Embedded Security in Cars USA(Ypsilanti, MI, USA). 15.

[34] Andrew Roberts. 2021. Self-Driving Vehicle Security Test Bed. Accessed 20 March. 2021. [Online]. Available: https://www.youtube.com/channel/UC7cXB9DSG6UCQAYHw4vkrSQ/videos

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on June 13,2025 at 16:07:42 UTC from IEEE Xplore. Restrictions apply.

Curriculum Vitae

1. Personal Data

Name	Andrew James Roberts
Date of Birth	26 July 1989
Nationality	Australia

2. Contact Information

Email	Andrew.Roberts@taltech.ee

3. Education

2020 - Present	*PhD Student (Cybersecurity)*. Tallinn University of Technology, Estonia.
2018 - 2020	*Master of Science (Cybersecurity)*. Tallinn University of Technology, Estonia.
2016 - 2018	*Master of Cybersecurity Operations*. University of New South Wales, Canberra, Australia.
2007 - 2011	*Bachelor of Information Technology/Bachelor of Arts*. Queensland University of Technology, Brisbane, Australia.

4. Professional Employment

2019 - Present	*Early Stage Researcher* Tallinn University of Technology

5. Language Competence

English	native

6. Field of Research

- 4.7. Information and Communications Technologies
- 4.8 Electrical Engineering and Electronics

7. Scientific Papers

- A. Roberts, L. Teply, M. Bellone, M. Pese, O. Maennel, M. Hamad, and S. Steinhorst. Fuzzsense: Towards a modular fuzzing framework for autonomous driving software.In arXiv, 2025.
- A. Roberts, M. Malayjerdi, M. Bellone, R. Sell, O. Maennel, M. Hamad, and S. Steinhorst. Adsecdata platform: An open-source data platform for autonomous driving cybersecurity. In 2025 IEEE 101st Vehicular Technology Conference (VTC2025-Spring), pages 1–7, 2025.
- A. Roberts, J. Cheng, O. Maennel, M. Hamad, and S. Steinhorst. Adseclang: A domain-specific language for cybersecurity testing of autonomous vehicles. In 2025 IEEE 101st Vehicular Technology Conference (VTC2025-Spring), pages 1–6, 2025.
- A. Roberts, M. Malayjerdi, M. Bellone, R. Sell, O. Maennel, M. Hamad, and S. Steinhorst. Analysis of autonomous driving software to low-level sensor cyber attacks. In 2025 IEEE/ACM 20th Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), pages 1–11, 2025.

- M. Hamad, A. Finkenzeller, M. Kühr, A. Roberts, O. Maennel, V. Prevelakis, and S. Steinhorst. React: Autonomous intrusion response system for intelligent vehicles. Computers Security, 145:104008, 2024.

- A. Roberts, M. R. H. Iman, M. Bellone, T. Ghasempouri, J. Raik, O. Maennel, M. Hamad, and S. Steinhorst. Adassure: Debugging methodology for autonomous driving control algorithms. In 2024 Design, Automation Test in Europe Conference Exhibition (DATE), pages 1–6, 2024.

- A. Roberts, M. Malayjerdi, M. Bellone, O. Maennel, and E. Malayjerdi. Analysing adversarial threats to rule-based local-planning algorithms for autonomous driving. Inaugural Symposium on Vehicle Security and Privacy (VehicleSec 2023) with NDSS, pages 1–8, 2023.

- A. Roberts, S. Marksteiner, M. Soyturk, B. Yaman, and Y. Yang. A global survey of standardization and industry practices of automotive cybersecurity validation and verification testing processes and tools. SAE International Journal of Connected and Automated Vehicles, 7, 11 2023.

- M. Malayjerdi, A. Roberts, O. M. Maennel, and E. Malayjerdi. Combined safety and cybersecurity testing methodology for autonomous driving algorithms. Proceedings of the 6th ACM Computer Science in Cars Symposium, pages 1–10, 2022.

- A. Roberts, O. Maennel, and N. Snetkov. Cybersecurity test range for autonomous vehicle shuttles. 2021 IEEE European Symposium on Security and Privacy Workshops (EuroSPW), pages 239–248, 2021.

Elulookirjeldus

1. Isikuandmed

Nimi	Andrew James Roberts
Sünniaeg	26 July 1989
Kodakondsus	Austraalia

2. Kontaktandmed

E-Post	Andrew.Roberts@taltech.ee

3. Haridus

2020 -...	*PhD tudeng (Cybersecurity)*. Tallinna Tehnikaülikool, Eesti
2018 - 2020	*Master of Science (Küberkaitse)*. Tallinna Tehnikaülikool, Eesti
2016 - 2018	*Master of Cybersecurity Operations*. University of New South Wales, Canberra, Austraalia.
2007 - 2011	*Bachelor of Information Technology/Bachelor of Arts*. Queensland University of Technology, Brisbane, Austraalia.

4. Teenistuskäik

2019 -...	*Doktorant-nooremteadur* Tallinna Tehnikaülikool

5. Keelteoskus

Inglise	emakeel

6. Teadustöö põhisuunad

- 4.7 Info-ja kommunikatsioonitehnoloogia
- 4.8 Elektrotehnika ja elektroonika

7. Akadeemilised Artiklid

- A. Roberts, L. Teply, M. Bellone, M. Pese, O. Maennel, M. Hamad, and S. Steinhorst. Fuzzsense: Towards a modular fuzzing framework for autonomous driving software.In arXiv, 2025.

- A. Roberts, M. Malayjerdi, M. Bellone, R. Sell, O. Maennel, M. Hamad, and S. Steinhorst. Adsecdata platform: An open-source data platform for autonomous driving cybersecurity. In 2025 IEEE 101st Vehicular Technology Conference (VTC2025-Spring), pages 1–7, 2025.

- A. Roberts, J. Cheng, O. Maennel, M. Hamad, and S. Steinhorst. Adseclang: A domain-specific language for cybersecurity testing of autonomous vehicles. In 2025 IEEE 101st Vehicular Technology Conference (VTC2025-Spring), pages 1–6, 2025.

- A. Roberts, M. Malayjerdi, M. Bellone, R. Sell, O. Maennel, M. Hamad, and S. Steinhorst. Analysis of autonomous driving software to low-level sensor cyber attacks. In 2025 IEEE/ACM 20th Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), pages 1–11, 2025.

- M. Hamad, A. Finkenzeller, M. Kühr, A. Roberts, O. Maennel, V. Prevelakis, and S. Steinhorst. React: Autonomous intrusion response system for intelligent vehicles. Computers Security, 145:104008, 2024.

- A. Roberts, M. R. H. Iman, M. Bellone, T. Ghasempouri, J. Raik, O. Maennel, M. Hamad, and S. Steinhorst. Adassure: Debugging methodology for autonomous driving control algorithms. In 2024 Design, Automation Test in Europe Conference Exhibition (DATE), pages 1–6, 2024.

- A. Roberts, M. Malayjerdi, M. Bellone, O. Maennel, and E. Malayjerdi. Analysing adversarial threats to rule-based local-planning algorithms for autonomous driving. Inaugural Symposium on Vehicle Security and Privacy (VehicleSec 2023) with NDSS, pages 1–8, 2023.

- A. Roberts, S. Marksteiner, M. Soyturk, B. Yaman, and Y. Yang. A global survey of standardization and industry practices of automotive cybersecurity validation and verification testing processes and tools. SAE International Journal of Connected and Automated Vehicles, 7, 11 2023.

- M. Malayjerdi, A. Roberts, O. M. Maennel, and E. Malayjerdi. Combined safety and cybersecurity testing methodology for autonomous driving algorithms. Proceedings of the 6th ACM Computer Science in Cars Symposium, pages 1–10, 2022.

- A. Roberts, O. Maennel, and N. Snetkov. Cybersecurity test range for autonomous vehicle shuttles. 2021 IEEE European Symposium on Security and Privacy Workshops (EuroSPW), pages 239–248, 2021.

ISSN 2585-6901 (PDF)
ISBN 978-9916-80-318-9 (PDF)