Cybersecurity Testbed for Connected and Autonomous Vehicles (CAVs)

Final Report

by

Satish V. Ukkusuri
Purdue University

Alvaro Cardenas, University of California, Santa Cruz (UCSC)
Daniel J. Fremont, UCSC
Gurcan Comert, Benedict College
Mansoureh Jeihani, Morgan State University
Mashrur Ronnie Chowdhury, Clemson University
M Sabbir Salek, Clemson University

June 2025

NATIONAL CENTER FOR TRANSPORTATION CYBERSECURITY AND RESILIENCY (TraCR)

DISCLAIMER

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated in the interest of information exchange. The report is funded, partially or entirely, by the National Center for Transportation Cybersecurity and Resiliency (TraCR) under Grant No. 69A3552344812 and 69A3552348317 which is headquartered at Clemson University, South Carolina, USA, from the U.S. Department of Transportation's University Transportation Centers Program. The U.S. Government assumes no liability for the contents or use thereof.

CONTACTS

For more information:

Satish V. Ukkusuri
G167D, 550 Stadium Mall Drive
West Lafayette, IN 47907, USA
Phone: 765-494-2296
Email: sukkusur@purdue.edu

TraCR
Clemson University
One Research Dr
Greenville, SC 29607
tracr@clemson.edu

ACKNOWLEDGMENT

This work is based upon the work supported by the National Center for Transportation Cybersecurity and Resiliency (TraCR), a U.S. Department of Transportation National University Transportation Center headquartered at Clemson University, Clemson, South Carolina, USA. Any opinions, findings, conclusions, and recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of TraCR. The U.S. Government assumes no liability for the contents or use thereof.

Technical Report Documentation Page

1. Report No. 2	**2. Government Accession No.** N/A	**3. Recipient's Catalog No.** N/A
4. Title and Subtitle Cybersecurity Testbed for Connected and Autonomous Vehicles (CAVs)		**5. Report Date:** June 2025 **6. Performing Organization Code:** N/A
7. Author(s) Satish V. Ukkusuri, Ph.D.; https://orcid.org/0000-0001-8754-9925 Alvaro Cardenas, Ph.D.; https://orcid.org/0000-0002-5142-9750 Daniel J. Fremont, Ph.D.; https://orcid.org/0000-0002-9992-9965 Mashrur "Ronnie" Chowdhury, Ph.D.; https://orcid.org/0000-0002-3275-6983 M. Sabbir Salek, Ph.D.; https://orcid.org/0000-0001-7326-3694 Gurcan Comert, Ph.D.; https://orcid.org/0000-0002-2373-5013 Mansoureh Jeihani, Ph.D.; https://orcid.org/0000-0001-8052-6931		**8. Performing Organization Report No.** 2
9. Performing Organization Name and Address National Center for Transportation Cybersecurity and Resiliency (TraCR), Clemson University, 414 A One Research Dr, Greenville, SC 29607 Purdue University, 610 Purdue Mall, West Lafayette, IN 47907 University of California, Santa Cruz, 1156 High St, Santa Cruz, CA 95064		**10. Work Unit No.** N/A **11. Contract or Grant No.** 69A3552344812 and 69A3552348317
12. Sponsoring Agency Name and Address U.S. Department of Transportation, Office of the Assistant Secretary for Research and Technology, 1200 New Jersey Avenue, SE, Washington, DC 20590		**13. Type of Report and Period Covered** Final Report, 01/01/2024 - 05/31/2025 **14. Sponsoring Agency Code** OST-R
15. Supplementary Notes Conducted under the U.S. DOT Office of the Assistant Secretary for Research and Technology's (OST-R) University Transportation Centers (UTC) program.		
16. Abstract The purpose of this project is to create a high-fidelity simulation platform to assess the cascading impacts of cyberattacks on connected and autonomous vehicle (CAV) systems. As cities increasingly adopt autonomous and connected technologies to enhance transportation efficiency and safety, they also expose critical infrastructure to new cyber-physical threats. While past studies have demonstrated vulnerabilities such as GPS spoofing and traffic signal manipulation, most are limited to small-scale scenarios and fail to capture system-wide consequences. We develop an open-source framework that supports scalable data generation, anomaly detection, and the simulation of complex attack scenarios across vehicle and network levels. Built on a co-simulation architecture, the platform integrates V2X communication, vehicle dynamics, and traffic flow models, and incorporates fleet control to evaluate operational disruptions. It also simulates basic safety message (BSM) attacks to reveal how compromised communications can trigger unsafe behaviors and traffic instabilities. By facilitating the study of realistic, large-scale threat scenarios and their cascading effects, this project lays a critical foundation for advancing transportation cybersecurity research and promotes the development of more resilient and secure mobility systems.		
17. Keywords Connected and Autonomous Vehicles (CAVs), Cyber-Physical Systems, Transportation Cybersecurity, Co-simulation Framework, V2X Communication, Large-Scale Cyberattack Simulation	**18. Distribution Statement** No restrictions.	

19. Security Classif. (of this report) Unclassified	**20. Security Classif. (of this page)** Unclassified	**21. No. of Pages** 44	**22. Price** N/A

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

TABLE OF CONTENTS

List of Tables

List of Figures

EXECUTIVE SUMMARY

As connected and autonomous vehicle technologies become increasingly integrated with transportation infrastructure and mobility services, cyberattacks can propagate beyond an individual vehicle or intersection and produce network- and fleet-level consequences. This project developed an open-source, multi-scale, high-fidelity CAV cybersecurity testbed whose principal contribution is systems integration. The platform combines METS-R SIM for network-scale traffic and fleet operations, CARLA for detailed vehicle and sensor simulation, Kafka for application-level V2X data streaming, and METS-R HPC for synchronized control through step, query, and control APIs. Preliminary interfaces to Scenic and Autoware further support structured scenario generation and full-stack autonomous-driving evaluation.

The testbed was exercised through five proof-of-concept attack classes: adversarial braking, BSM replay and spoofing, route guidance manipulation, adversarial booking, and denial of service. A Town05 co-simulation represented approximately 100 trips per simulated minute, totaling 1,000 trips over 10 simulated minutes, and support was extended to Town01–Town07 and Town10. In a physical C-V2X DoS experiment, receiver latency increased above 500 packets per second, and packet drops occurred at 1,000 packets per second. Other experiments demonstrated the propagation of braking disturbances, queue-wide responses to falsified safety messages, route deviations, and fleet clustering under fake passenger demand

The project provides publicly available simulation tools, interactive attack notebooks, and demonstration resources that can support transportation cybersecurity research, attack reproduction, and defense evaluation. Future work includes full communication-network simulation (e.g., NS-3) integration, automated and data-driven scenario synthesis, systematic quantitative validation across repeated runs, and broader hardware-in-the-loop testing.

CHAPTER 1
Introduction

1.1 Background

Enhancing transportation systems through autonomy and connectivity has been a top priority for major car manufacturers and technology companies. While new technologies and algorithms hold promises in promoting efficiency and safety, they also introduce vulnerabilities. Previous research has demonstrated viable attacks on connected and autonomous vehicles (CAVs), such as GPS spoofing (Zeng et al., 2018) and tactics involving the manipulation of traffic signals (Chen et al., 2018; Yan et al., 2022). However, most studies are based on small-scale scenarios (e.g., one vehicle, one intersection, or one link), which can only reflect the local and limited impact of the attacks. The lack of a simulator that can reliably test cybersecurity features (i.e., attacks and defenses) presents three-fold challenges: traditional traffic simulators themselves cannot simulate CAV-specific features like V2X communication; photo-realistic simulators cannot cover large-scale scenarios; and the absence of APIs for verifying attacks and defenses. To comprehensively evaluate the threats associated with cyberattacks against CAVs and assess the effectiveness of defense mechanisms, a faithful simulation testbed capable of handling multi-scale system dynamics is needed.

In this project, we aim to develop a sophisticated simulation testbed capable of assessing the multi-scale impact of cyberattacks against CAV systems. Unlike existing simulators (Lopez et al., 2018; Zhang et al., 2023), we adopt a co-simulation framework to model multi-scale system dynamics from V2X communication, vehicle maneuvering, and car-following to vehicle scheduling, routing, and network-level congestion effects. The ultimate goal is to construct a reliable environment that can serve as a foundational platform for future cybersecurity studies.

The simulation is developed to fit the NIST risk assessment process (Ross, 2012), shown in Figure 1. It is envisioned that this new simulation testbed will enable more effective vulnerability identification, attack validation, and understanding of potential attack effects. Finally, our simulator can facilitate the development of more robust and secure CAV systems by identifying potential vulnerabilities and testing various defense strategies.

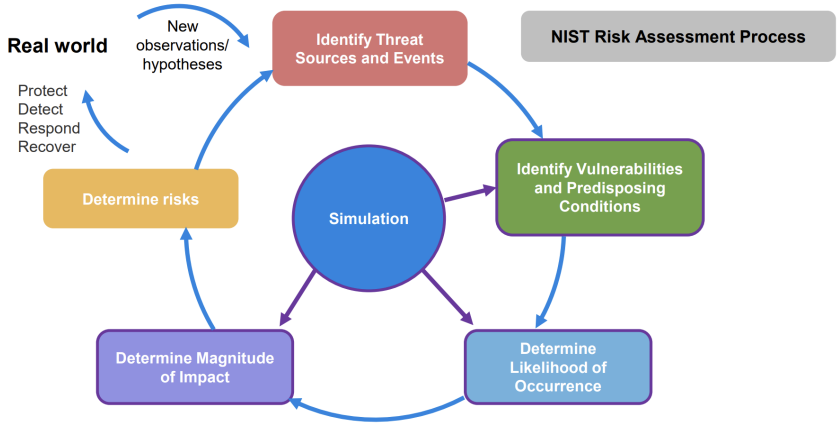

Figure 1: The NIST risk assessment process

1.2 Project objectives

The overarching goal of this project is creating a **virtual testbed** that enables monitoring of existing vulnerabilities, assessing their risks, and testing diverse attack/defense algorithms. We also expect that our numerical tests can forge a better understanding of the potential risks of CAVs among policymakers and the public. To achieve this goal, we plan to address four objectives:

1. Capturing the cascading effect of cyberattacks in transportation networks: We will explore how cyberattacks can trigger cascading effects within interconnected road traffic systems. Our focus is on understanding the extent of possible disruptions, identifying critical points of vulnerability, and creating strategies for reducing these cascading effects.

2. Simulation language for cyberattacks: This mechanism allows us to create, simulate, and analyze diverse cyberattack scenarios targeting CAVs. Through scenario generation, we can replicate various cyber threats, ranging from GPS spoofing to malware intrusions, to assess their potential impact on CAV fleets.

3. Verifying common attacks: We will use our testbed to simulate common attacks on CAV fleets, like GPS spoofing, data falsification, and malware intrusions. By studying the tangible effects of these attacks on our experimental setup, we can better comprehend their real-world implications, validate theoretical models, and develop effective defense mechanisms.

4. Modeling traffic flow dynamics under cyberattacks: This objective aims to model and then predict the influence of cyberattacks on the dynamic behavior of traffic. This modeling will enable us to obtain early warnings for potential disruptions, such as increased congestion resulting from altered travel routes due to cyberattacks.

1.3 Organization of this report

This rest of this document is organized as follows:

- Chapter 2 provides the related work on traffic simulation, security of connected and autonomous vehicles.
- Chapter 3 focuses on developing the simulation framework capable of capturing the lifecycle impacts of cyberattacks.
- Chapter 4 reports the V2X data stream simulation based on Kafka, a widely used event streaming platform in real-time data processing applications. The team also explored the viability of integrating this framework with NS-3, which may enhance realism in simulating the communication delay.
- Chapter 5 details the APIs for interactively controlling the data stream, decision, and states in the simulation.
- Chapter 6 presents the co-simulation implementation between CARLA and METS-R SIM, with an explorative effort on integrating with Scenic, a probabilistic programming system for environment modeling (world modeling) and data generation to facilitate the discovery of the riskiest scenarios.
- Chapter 7 shows the details of validating the simulation functionality, with openly accessible demo available at project repos[1].
- Chapter 8 concludes this project and discusses future directions.

[1] METS-R SIM: https://github.com/umnilab/METS-R_SIM
METS-R HPC: https://github.com/umnilab/METS-R_HPC

CHAPTER 2

Literature Review

2.1 Traffic simulators for CAVs

With the advent of connected and autonomous vehicles (CAVs), traditional traffic simulators have evolved to accommodate the needs of modeling these systems. Traffic simulators for CAVs aim to model not only the vehicle flow and traffic dynamics but also the interactions enabled by V2X (Vehicle-to-Everything) communication and the complex behaviors introduced by autonomous driving technologies. One of the prominent simulators in this domain is the Simulation of Urban MObility (SUMO), which has been adapted to support CAV functionalities through extensions such as SUMO-V2X. SUMO-V2X integrates vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication models, allowing for the simulation of cooperative driving strategies and dynamic traffic management. SUMO-V2X enables detailed simulation of CAV interactions by leveraging communication protocols that allow vehicles to exchange information about their positions, speeds, and intended maneuvers. This integration enhances the accuracy of simulations in reflecting real-world driving conditions and traffic behaviors. Additionally, the SUMO-based Hardware-in-the-Loop (HIL) simulation framework facilitates the testing and rapid prototyping of cooperative vehicular applications (Szendrei et al., 2018). Another notable simulator is PTV VISSIM, which has incorporated features for modeling CAV behavior and V2X communication. Recent enhancements to VISSIM include modules for simulating CAV-specific interactions and network effects. One significant enhancement is the inclusion of modules that support the simulation of CAV-specific interactions. VISSIM now offers tools for simulating network effects, which are essential for assessing the broader impact of CAVs on transportation networks (Vrbanić et al., 2021).

The Advanced Driver Assistance Systems (ADAS) Simulator is another example that integrates CAV-specific features, including sensor simulation and real-time vehicle control. For instance, Ansys AVxcelerate provides a comprehensive platform for physically accurate sensor simulations involving cameras, LiDAR, radar, and thermal sensors. This platform supports hardware-in-the-loop (HIL) and software-in-the-loop (SIL) testing, enabling the development and validation of sensor perception systems in real-time (Hussein et al., 2018). Additionally, Bosch Engineering has also developed an ADAS simulator based on the Unreal Engine, which supports the development and validation of systems like Adaptive Cruise Control (ACC). This simulation environment allows for software-in-the-loop testing, providing a realistic virtual setting for demonstrating, calibrating, and analyzing ADAS features without the need for an actual vehicle or electronic control unit (ECU) [2] . These simulators offer detailed insights into vehicle-to-vehicle communications and the impact of autonomous driving on traffic flow. However, they often face limitations such as scalability and integration with real-world data.

In addition to these, the co-simulation environment for Autonomous Driving (CoSim-AD) provides a framework for integrating traffic simulation with autonomous vehicle control systems. CoSim-AD enables the evaluation of complex interactions between autonomous vehicles and traffic infrastructure on a large scale (Cantas and Guvenc, 2023; Li et al., 2021a). Despite these

[2] https://www.bosch-engineering.com/stories/stories-detailpages/t-storypage-12.html

advancements, many simulators still struggle with integrating large-scale, multi-modal interactions and ensuring comprehensive coverage of all relevant scenarios.

2.2 Operation modules in CAV systems

Key operation modules in CAV systems include Vehicle-to-Everything (V2X) communication, sensor fusion, autonomous control algorithms, and decision-making systems. V2X communication is a foundational component of CAV systems, enabling vehicles to exchange information with other vehicles, infrastructure, and network services. This capability enhances situational awareness and coordination, facilitating safer and more efficient driving. (Gelbal et al., 2023) and (Ribeiro et al., 2023) demonstrated that using V2X communication technology significantly reduces collision risks and congestion by increasing the prediction rate and interaction ability using real-time data of vehicle information and machine learning technology. (Li et al., 2024) found that integrating V2X communication technologies in traffic management systems improves computational efficiency by 23% and reduces the collision rate by 13%. The V2X ecosystem includes various communication types, such as V2V, V2I, and V2P (Vehicle-to-Pedestrian), each addressing different aspects of vehicular communication. V2V communication allows vehicles to directly exchange information about their speed, position, and direction, which helps in collision avoidance and cooperative driving (Ngo et al., 2023). V2I communication connects vehicles with roadside infrastructure, such as traffic lights and road signs, to optimize traffic flow and manage traffic congestion. V2I communication enables vehicles to receive traffic signal information and adjust their speed, accordingly, reducing waiting times at intersections and improving fuel efficiency (Khan et al., 2022). V2P communication addresses the safety of pedestrians by allowing vehicles to communicate with pedestrian devices, such as smartphones or wearable devices. This type of communication can alert drivers to the presence of pedestrians in or near the roadway, enhancing pedestrian safety (Malik et al., 2020).

Sensor fusion integrates data from multiple sensors, including cameras, radar, and LiDAR, to create a comprehensive representation of the vehicle's environment. This fusion process is critical for the accurate perception and understanding of surrounding conditions, which in turn informs decision-making and control processes. Studies have highlighted the importance of sensor fusion in overcoming the limitations of individual sensors. Cameras provide detailed visual information but struggle with depth perception and poor lighting conditions. LiDAR offers accurate distance measurements but can be affected by weather conditions. Radar is robust in various environmental conditions but lacks the high-resolution imaging of cameras and LiDAR (Fayyad et al., 2020; Huang et al., 2022). Additionally, some studies demonstrated that using deep learning-based approaches has significantly improved sensor fusion techniques. These methods leverage the strengths of each sensor type to produce a more accurate and reliable perception of the environment (Zhuang et al., 2023; Li et al., 2022). The integration of sensor data allows for redundancy, which is crucial for safety and reliability. If one sensor fails or provides inaccurate data, the system can still rely on other sensors to maintain accurate perception and control of the vehicle (Yeong et al., 2021). Effective sensor fusion relies on advanced algorithms to combine data from disparate sources and handle uncertainties.

Autonomous control algorithms govern the vehicle’s behavior, including path planning, obstacle avoidance, and adaptive cruise control. These algorithms utilize data from sensors and V2X

communication to make real-time decisions and control the vehicle's movements. Some studies highlight the integration of artificial potential field algorithms for optimal path planning. These algorithms guide autonomous vehicles by calculating desired trajectories while avoiding obstacles, using sensor data to dynamically adjust the vehicle's path in real time (Park and Choi, 2023). Another study focuses on the methods used for obstacle avoidance and path planning in autonomous navigation. It discusses traditional approaches as well as more recent developments in genetic algorithms and neural network-based methods. These techniques help vehicles navigate their environment safely, avoiding collisions and efficiently reaching their destinations (Oroko and Nyakoe, 2022). The development of robust control algorithms is essential for ensuring the safety and reliability of autonomous driving.

Decision-making systems in CAVs involve high-level planning and strategy formulation, including route optimization and behavior prediction. These systems use predictive models and machine learning techniques to anticipate and respond to dynamic traffic conditions and potential hazards. Some studies discuss how decision-making systems in CAVs use algorithms such as rapidly exploring random tree and its variants for path planning at intersections. These algorithms help generate optimal paths by considering the road geometry and potential obstacles, enhancing the vehicle's ability to navigate complex environments safely (Li et al., 2021b). Another research emphasizes the integration of predictive models, including neural networks and recurrent neural networks, to forecast the behavior of surrounding vehicles. This predictive capability allows CAVs to make informed decisions in real time, significantly improving safety and efficiency (HomChaudhuri and Bhattacharyya, 2022). Moreover, AI-driven decision-making systems are essential for route optimization and behavior prediction. These systems leverage vast amounts of data from sensors and V2X communication to continuously update and optimize routes based on real-time traffic conditions (Garikapati and Shetiya, 2024).

Collectively, these operation modules work together to enable the functionality of CAVs. However, their integration presents challenges in terms of complexity, data management, and cybersecurity, underscoring the need for comprehensive simulation tools to evaluate their performance and interactions effectively. A review paper (Parekh et al., 2022) shows that simulations are valuable for testing, they often fail to capture the full complexity of real-world scenarios, leading to gaps between simulated performance and actual performance on the road. Some other studies have shown that large networks with numerous CAVs often require simplifications that can reduce the accuracy and applicability of the results and better validation techniques are needed to ensure that simulation results are reliable (Rana and Hossain, 2023).

2.3 Summary of literature

While existing simulators such as SUMO, VISSIM, and CoSim-AD have advanced CAV modeling by incorporating V2X communication and autonomous behaviors, they remain limited in scalability, realism, and the ability to simulate coordinated cyberattacks. Most prior studies evaluate cyber threats in isolated, small-scale settings, failing to capture system-wide, cascading impacts. Additionally, current simulations often oversimplify large-scale dynamics and lack mechanisms for validating attack and defense strategies. These gaps underscore the need for a comprehensive, high-fidelity simulation platform that supports multi-scale modeling and robust cybersecurity testing for CAV systems.

CHAPTER 3

Simulation Framework

3.1 Framework design

The overall framework of our simulation is shown in Figure 2. Within this framework, we utilize the METS-R SIM (Lei et al., 2024) for large-scale multi-modal traffic simulation alongside CARLA (Dosovitskiy et al., 2017) for photo-realistic sensors and driving simulations. This pairing provides us with comprehensive testing environments. In addition to the traffic simulators, a Kafka-based component is introduced to explicitly model the data stream, exposing it to potential attacks such as Denial-of-Service (DoS) and data spoofing. Moreover, this setup enables the separation of logic for data processing, decision generation, traffic simulation, and attacks. Finally, the remote controller serves as the mediator between multiple simulation instances and the data stream. This framework guarantees strict synchronization. By broadcasting simulation ticks among the simulator, data processor, and attacker, the controller ensures consistent testing of attack algorithms with any level of computational costs.

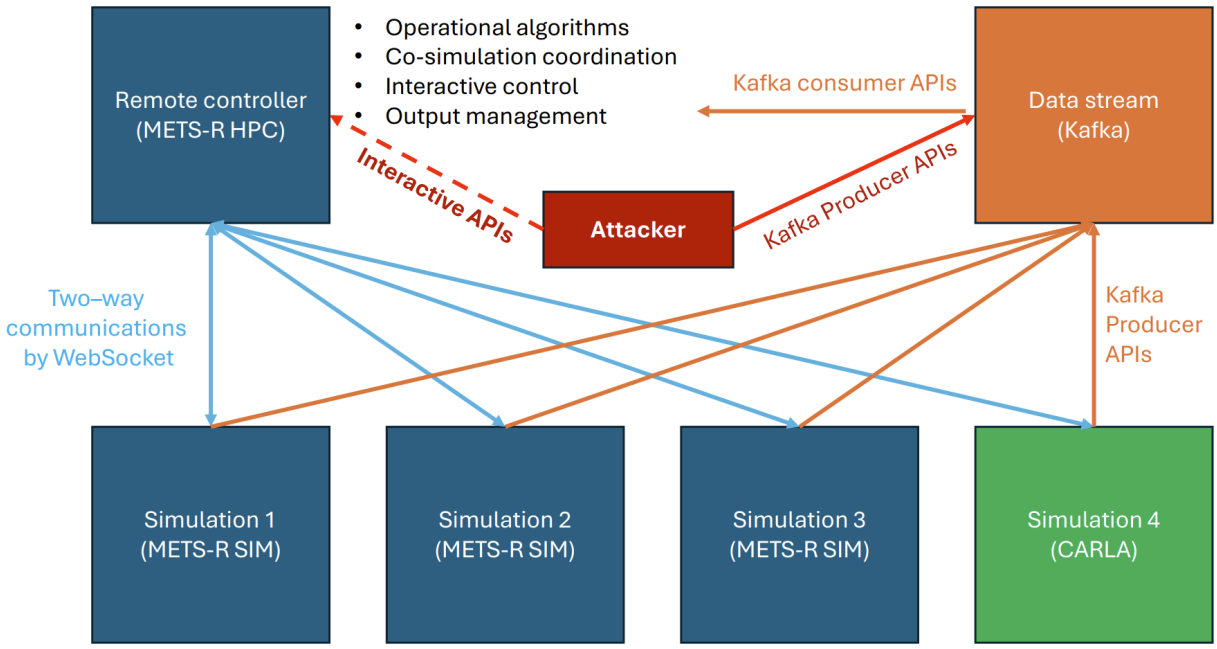

Figure 2: Simulation framework

3.2 Remote controller

The remote controller aims to model the data collection and processing workflow faithfully in CAV applications. Functionally, it acts as the mediator of different simulators and data streams. To develop this module, the team extends the high-performance-computing (HPC) module in the METS-R SIM platform (Lei et al., 2024) by adding functions to interact with data stream and

control APIs that will be described in detail in the following sections. Additionally, the team integrates basic anomaly detection module for time series data using the anomaly detection algorithms provided in a Python package named ADTK[3].

3.3 Kafka-based data stream

The team uses Kafka to explicitly model data stream. Three data streams have been implemented and tested. They are the basic safety message (BSM), the link travel time, and the link energy consumption. The BSM data stream is generated by connected vehicles using vehicular sensors and communicated via dedicated short-range communication (DSRC) or C-V2X. For our simulating purposes, we record the actual location of the data generator to capture the differences in communication ranges between these two technologies. The latter two data streams are generated to mimic the data collected by mapping apps and connected car companies.

3.4 Interactive APIs

To facilitate attacks/defense assessment, the team develops three types of APIs in the remote controller.

- Step API: This API comprises one function to advance the simulation and one variable to track the current tick of the simulation. This will enable co-simulation with other simulators and ensure timely execution of generated attacks.
- Query APIs: These APIs enable the controller to gather the state of agents in METS- R SIM (e.g., vehicles, roads, lanes, signals, demand zones, charging stations) during the simulation.
- Control APIs: These APIs allow us to alter vehicle locations and acceleration/lane-changing decisions during the simulation, which will be utilized in co-simulation development.

3.5 Co-simulation

A co-simulation module is developed to address two key applications: Exploring Cascading Effects of Attacks on CAVs and Controlling Ego-Vehicle with Computer Vision Inputs. The first application tackles the challenge of CARLA's computational expense by using a small sub-network to simulate a link or intersection, which generates detailed sensor data affecting traffic controls, such as signals and vehicle routes, within METS-R SIM. The second application utilizes CARLA to implement autonomous driving algorithms (e.g., Autoware) by simulating the same network in both CARLA and METS-R SIM. CARLA produces vehicle control commands that are translated into acceleration and speed inputs for METS-R SIM, facilitating interaction between the simulators.

[3] https://adtk.readthedocs.io/en/stable/

CHAPTER 4

Kafka-based V2X communication modeling

4.1 Kafka-based data stream

Apache Kafka is an open-source distributed event streaming platform widely used in commercial applications for real-time data processing, including banking, logistics, and automotive systems. In our simulation framework, we adopt Kafka to model the V2X (Vehicle-to-Everything) communication layer due to its scalability, reliability, and ability to handle high-throughput data streams. Kafka enables asynchronous, loosely coupled communication between simulation modules, supporting both real-time analysis and logging of sensor data. It also facilitates flexible integration of attack and defense mechanisms that interact with the data stream, making it well-suited for simulating CAV cybersecurity threats.

4.2 Data streams

Three data streams have been implemented and tested, including: the basic safety message (BSM), the link travel time, and the link energy consumption. The BSM data stream is generated by connected vehicles using vehicular sensors and communicated via dedicated short-range communication (DSRC) or C-V2X. The latter two data streams are generated to mimic the data collected by mapping apps (such as Google Maps) and connected car platforms.

4.2.1 Basic safety message

The BSM data stream is generated by connected vehicles using vehicular sensors and communicated via dedicated short-range communication (DSRC) or C-V2X. For our simulating purposes, we record the actual location of the data generator to capture the differences in communication ranges between these two technologies.

Table 1: Basic safety message fields

Field Name	Description
vid	Vehicle ID
utc_fix_mode	GPS fix status
latitude, longitude, altitude	Geographic position
qty_SV_in_view / qty_SV_used	Number of satellites in view / used for fix
GNSS_unavailable	GPS status flag
GNSS_aPDOPofUnder5	Accuracy condition flag
GNSS_inViewOfUnder5	Satellite availability flag
GNSS_localCorrectionsPresent	Indicates local correction data is present
GNSS_networkCorrectionsPresent	Indicates network correction data is present
SemiMajorAxisAccuracy	Horizontal positioning uncertainty
SemiMinorAxisAccuracy	Vertical positioning uncertainty

Field Name	Description
SemiMajorAxisOrientation	Orientation of the confidence ellipse
heading, velocity, climb	Motion data
time_confidence, velocity_confidence, elevation_confidence	Confidence measures for timing, speed, and altitude
leap_seconds	GPS-to-UTC offset
utc_time	Timestamp
type	Communication type (0 = DSRC, 1 = C-V2X)
true_x, true_y	Ground-truth location (used internally for visibility modeling only)

4.2.2 Link travel time

The link travel time stream simulates the data collected by mobile navigation apps and mapping platforms (e.g., Google Maps). Each message reports the time taken by a vehicle to traverse a specific road segment at a given point in the simulation.

Table 2: Link travel time fields

Field Name	Description
tick	Simulation time when the data was recorded
vid	Vehicle ID
roadid	Unique identifier for the road segment
veh_type	Vehicle type (e.g., passenger car, truck, etc.)
road_length	Length of the road (meters)
traveltime	Time taken to travel the road segment (seconds)
utc_time	Timestamp in UTC format

4.2.3 Link energy consumption

The link energy consumption stream represents data similar to that gathered by connected vehicle platforms, providing energy usage statistics for each road segment a vehicle traverses.

Table 3: Link energy consumption fields

Field Name	Description
vid	Vehicle ID
vehType	Vehicle type (e.g., electric, hybrid, etc.)
roadID	Identifier for the road segment
linkEnergy	Energy consumed over the road segment (in kWh or equivalent unit)
utc_time	Timestamp in UTC format

4.3 Implementation details

Kafka and supporting services are deployed in Docker containers for ease of development and reproducibility. A PostgreSQL database is also initialized to store stream data persistently, enabling queries for historical analysis and decision-making during attack or defense evaluation. On the simulation side, a Kafka producer service is activated with each run, continuously collecting data from vehicles equipped with relevant sensor types:

- DSRC / C-V2X for BSMs
- MOBILEDEVICE for travel time and energy consumption

This modular structure supports real-time streaming, fault injection, and tight integration with the rest of the simulation framework.

4.4 C-V2X communication with NS-3

The team has explored the integration of **NS-3**, a widely used discrete-event network simulator, into our testbed to enhance the realism of V2X communication modeling, particularly with respect to **latency, packet loss, and transmission reliability** in connected vehicle environments. This enhancement is needed to accurately reflecting the timing and reliability constraints of DSRC and C-V2X systems, which are critical to the safety and responsiveness of autonomous vehicle operations.

Our evaluation confirms that such integration is technically feasible. The workflow involves generating structured input files that define communication scenarios based on vehicle locations, message types, and transmission parameters. These inputs, as illustrated by the example in Figure 3, are then passed to an externally executed instance of NS-3. Upon completion, the simulation returns key metrics such as transmission latency, packet delivery status, and other relevant network statistics that are measurable through Wireshark. These outputs can be reintegrated into the main simulation loop, enabling co-simulation that reflects realistic network-induced delays and disruptions.

```
$node_(0) set X_ 715.3
$node_(0) set Y_ 34.35
$node_(0) set Z_ 0
$ns_ at 0.0 "$node_(0) setdest 715.3 34.35 0.00"
$ns_ at 1.0 "$node_(0) setdest 713.68 34.32 1.63"
$node_(1) set X_ 274.37
$node_(1) set Y_ 60.44
$node_(1) set Z_ 0
$ns_ at 1.0 "$node_(1) setdest 274.37 60.44 0.00"
$ns_ at 2.0 "$node_(0) setdest 710.42 34.24 3.27"
$node_(2) set X_ 282.37
$node_(2) set Y_ 65.82
$node_(2) set Z_ 0
$ns_ at 2.0 "$node_(2) setdest 282.37 65.82 0.00"
$ns_ at 3.0 "$node_(0) setdest 705.44 34.13 5.00"
$ns_ at 3.0 "$node_(2) setdest 281.64 66.77 1.51"
$node_(3) set X_ 199.87
$node_(3) set Y_ 218.66
$node_(3) set Z_ 0
$ns_ at 3.0 "$node_(3) setdest 199.87 218.66 0.00"
$ns_ at 4.0 "$node_(0) setdest 698.69 33.98 6.77"
$ns_ at 4.0 "$node_(2) setdest 279.84 69.24 3.06"
$ns_ at 4.0 "$node_(3) setdest 201.51 219.57 1.88"
$node_(4) set X_ 212.24
$node_(4) set Y_ 235.2
$node_(4) set Z_ 0
$ns_ at 4.0 "$node_(4) setdest 212.24 235.2 0.00"
$ns_ at 5.0 "$node_(0) setdest 689.97 33.78 8.75"
$ns_ at 5.0 "$node_(2) setdest 277.2 72.83 4.46"
$ns_ at 5.0 "$node_(3) setdest 204.92 221.46 3.90"
$ns_ at 5.0 "$node_(4) setdest 210.93 234.09 1.75"
$node_(5) set X_ 279.28
$node_(5) set Y_ 55.61
$node_(5) set Z_ 0
```

Figure 3: Example inputs for NS-3

CHAPTER 5

Interactive APIs development

The remote controller supports a variety of APIs that allow simulation clients, co-simulation components, and attackers to interact with the simulation in a flexible and controlled manner. These APIs are grouped into three categories: Step API, Query APIs, and Control APIs. Together, they enable synchronized progression of simulation time, real-time data extraction, and direct control over vehicle behavior and mobility operations.

5.1 Step API

The Step API controls the progression of the simulation by advancing it one tick at a time. This API is essential for synchronizing events across co-simulation modules and ensuring deterministic execution during attack scenario testing.

Table 4: Step API

Name	Description	Examples	Inputs
tick	Move the simulation n ticks forward	sim_client.tick(n)	n

5.2 Query APIs

The Query APIs allow users to retrieve real-time information from the simulation, including the states of vehicles, roads, signals, zones, and charging infrastructure. These APIs support both high-level monitoring and targeted attack/defense interventions by exposing internal simulation states.

Table 5: Query APIs

Name	Description	Examples	Return
query_vehicle(id=None, private_veh=False, transform_coords=False)	Get the information of one or more vehicles. By default, queries public vehicles. Set *private_veh=True* for private vehicles and *transform_coords=True* to convert coordinates to SUMO format.	sim_client.query_vehicle(1, True, True)	x, y, speed, acceleration, bearing, origin zone, destination zone, vehicle state
query_bus(id=None)	Get the information of one or more EV buses.	sim_client.query_bus(37732)	battery state, route ID, current stop, passenger number
query_taxi(id=None)	Get the information of one or more EV taxis.	sim_client.query_taxi(36879)	x, y, passenger number, vehicle state, origin zone, destination zone
query_road(id=None)	Get the information of one or more roads.	sim_client.query_road(100011)	speed limit, number of on-road vehicles, average travel time, road length (m), energy consumed, road type

query_zone(id=None)	Get the information of one or more demand zones.	sim_client.query_zone(0)	vehicle stock, taxi demand, bus demand, x, y, zone type
query_signal(id=None)	Get the information of one or more traffic signals.	sim_client.query_signal(14)	current phase, next phase, next update time
query_chargingStation(id=None)	Get the information of one or more EV charging stations.	sim_client.query_chargingStation(-10)	x, y, number of available chargers
query_coSimVehicle()	Get the vehicle IDs currently located on the co-simulation road.	sim_client.query_coSimVehicle()	list of vehicle IDs on the co-sim road

5.3 Control APIs

The Control APIs allow users to intervene in the simulation by controlling vehicle behavior, modifying routes, issuing service requests, or resetting the simulation state. These APIs are essential for implementing attack strategies, testing defense mechanisms, or conducting scenario manipulation in co-simulation experiments.

Table 6: Control APIs

Name	Description	Examples	Inputs
generate_trip(vehID, origin=-1, destination=-1)	Generate a vehicle trip between origin and destination zones.	sim_client.generate_trip("veh_1", 3, 5)	vehID, origin, destination
generate_trip_between_roads(vehID, origin, destination)	Generate a vehicle trip between origin and destination roads.	sim_client.generate_trip_between_roads("veh_1", "r1", "r2")	vehID, origin, destination
set_cosim_road(roadID)	Set one or more roads for co-simulation.	sim_client.set_cosim_road("road_2")	roadID
release_cosim_road(roadID)	Release one or more co-simulation roads.	sim_client.release_cosim_road("road_2")	roadID
teleport_cosim_vehicle(vehID, x, y, bearing, private_veh=False, transform_coords=False)	Teleport a vehicle to a location using coordinates.	sim_client.teleport_cosim_vehicle("veh_1", 10.5, 25.0)	vehID, x, y, bearing, private_veh, transform_coords
exit_cosim_region(vehID, x, y, private_veh=False, transform_coords=False)	Remove a vehicle from the co-simulation region and add it to the road based on the assigned coordinates.	sim_client.exit_cosim_region("veh_1", 10.5, 25.0)	vehID, x, y, private_veh, transform_coords
teleport_trace_replay_vehicle(vehID, roadID, laneID, dist, private_veh=False)	Teleport a vehicle using road ID, lane ID, and distance from downstream junction.	sim_client.teleport_trace_replay_vehicle("veh_1", "road_3", 1, 30.0)	vehID, roadID, laneID, dist, private_veh
enter_next_road(vehID, private_veh=False)	Move vehicle to next road in its planned route.	sim_client.enter_next_road("veh_1")	vehID, private_veh
control_vehicle(vehID, acc, private_veh=False)	Directly control vehicle acceleration.	sim_client.control_vehicle("veh_1", 2.5)	vehID, acc, private_veh

update_vehicle_sensor_type(vehID, sensorType, private_veh=False)	Change the sensor type of a vehicle.	sim_client.update_vehicle_sensor_type("veh_1", "LiDAR")	vehID, sensorType, private_veh
dispatch_taxi(vehID, orig, dest, num)	Dispatch a taxi between two zones.	sim_client.dispatch_taxi("taxi_1", 0, 2, 1)	vehID, orig, dest, num
dispatch_taxi_between_roads(vehID, orig, dest, num)	Dispatch a taxi between roads.	sim_client.dispatch_taxi_between_roads("taxi_1", "r1", "r2", 1)	vehID, orig, dest, num
add_taxi_requests(zoneID, dest, num)	Add taxi passenger request at a zone.	sim_client.add_taxi_requests(1, 3, 2)	zoneID, dest, num
add_taxi_requests_between_roads(zoneID, orig, dest, num)	Add road-based taxi request from a zone.	sim_client.add_taxi_requests_between_roads(1, "r1", "r2", 1)	zoneID, orig, dest, num
assign_request_to_bus(vehID, orig, dest, num)	Assign passenger request to a specific bus.	sim_client.assign_request_to_bus("bus_1", 0, 5, 10)	vehID, orig, dest, num
add_bus_requests(zoneID, dest, num)	Add a bus passenger request at a zone.	sim_client.add_bus_requests(2, 5, 15)	zoneID, dest, num
reset(prop_file)	Reset simulation using a specified property file.	sim_client.reset("Data.properties.NYC")	prop_file
reset()	Reset simulation.	sim_client.reset()	map_name
terminate()	Fully terminate the simulation and close the client.	sim_client.terminate()	None
close()	Close the client while leaving the simulation running.	sim_client.close()	None

5.4 Implementation details

5.4.1 Simulation side (JAVA program)

On the simulation side, the system uses modular message-handling architecture implemented in Java, built around two core abstract classes: MessageHandler and MessageSender.

- **MessageHandler** is designed to process incoming WebSocket messages. It maintains a registry of message types and their corresponding handler functions using a Map<String, CustomizableHandler>. Each handler processes JSON messages (JSONObject) and returns structured results. The handleMessage() method serves as the main dispatch logic for decoding and routing incoming requests based on their type.
- **MessageSender** manages outbound communication to remote clients over WebSockets. It verifies connection status and transmits serialized messages, tracking the count of messages sent. The sender ensures robust message delivery by validating session readiness before sending.

This architecture ensures clean separation between message reception and message processing, supports extensibility for new message types, and facilitates asynchronous communication with the remote controller.

5.4.2 Remote controller side (Python program)

The remote controller is implemented in Python using WebSocket-based communication. It is responsible for issuing control commands, querying simulation states, managing co-simulation logic, and orchestrating attacks or interventions.

The core class, METSRClient, handles:

- **Connection Management**: The client attempts to establish a WebSocket connection to a specific simulation server address and retries if the connection fails. It waits for a readiness acknowledgment before proceeding.
- **Message Handling**: Using methods like send_msg(), receive_msg(), and send_receive_msg(), the client sends JSON-encoded control or query messages and waits for structured responses from the server. Thread locking ensures synchronized access in multi-threaded scenarios.
- **Tick Synchronization**: The tick() method advances the simulation by a specified number of ticks and waits for a confirmed update, ensuring alignment with external processes (e.g., co-simulated agents or anomaly detectors).
- **Query Functions**: A suite of query_*() methods retrieves real-time simulation data about vehicles, roads, signals, zones, charging stations, and co-simulation entities. Each method supports single or batch queries, automatically formatting messages as needed.
- **Control Functions**: Control methods such as generate_trip(), teleport_cosim_vehicle(), dispatch_taxi(), and control_vehicle() allow users or algorithms to intervene in the simulation, modify trajectories, or simulate adversarial actions. These functions all follow a common pattern of message construction, transmission, and response validation.
- **Simulation Reset and Termination**: The client supports full simulation resets via reset() and graceful shutdown via terminate() and close() methods. These ensure the state is reinitialized and resources are properly released.
- **Visualization Integration**: When enabled, the controller can launch or terminate a visualization server tied to the latest simulation output for monitoring and debugging purposes.

Together, these implementations enable flexible, robust, and high-performance co-simulation between METS-R SIM and other external modules such as CARLA or machine learning agents. The clear API structure ensures reproducibility, supports fault injection, and enables rapid development of testing pipelines for connected and autonomous vehicle cybersecurity research.

CHAPTER 6

Co-simulation development

6.1 Applications scenarios

The team formalizes two target applications for co-simulation:

- **Exploring the cascading effects of attacks on CAVs**: This application addresses the issue that CARLA is a computationally expensive platform and cannot efficiently handle large-scale (e.g., city-level) simulation problems. In this application, we have a large network represented in METS-R SIM, with a small sub-network (e.g., a link or one intersection) simulated in CARLA to produce comprehensive sensor data, including camera and LiDAR data. This data subsequently influences controls such as traffic signals and vehicle routes, with their broader effects, which cannot be efficiently simulated in CARLA, captured within METS-R SIM.
- **Controlling ego-vehicle with computer vision inputs:** This application integrates CARLA as an interface for autonomous driving software stacks (e.g., Autoware). The same traffic network is mirrored in both CARLA and METS-R SIM. CARLA is responsible for simulating the ego vehicle, which processes sensor inputs and outputs low-level control commands (throttle, brake, and steering). These commands are translated into speed and acceleration inputs for METS-R SIM. To maintain efficiency, only the ego vehicle and its immediate surroundings are rendered in CARLA.

6.2 Implementing the co-simulation functions

We have developed a synchronization mechanism that coordinates simulation steps between METS-R SIM and CARLA. To address communication latency and avoid unnecessary overhead, we implement a **lazy update mechanism** that limits data exchange to essential updates only.
As a proof of concept for the first application, we created a co-simulation demo using CARLA Town05, where CARLA simulates a single road and METS-R SIM manages the rest of the network. The simulation produces approximately 100 trips per minute, totaling 1,000 vehicle trips over a 10-minute period. The demo successfully demonstrates signal-following behavior by the CARLA-controlled vehicle based on traffic light states managed by METS-R SIM. We have also added support for co-simulation across all official CARLA maps (Town01 to Town07, and Town10).

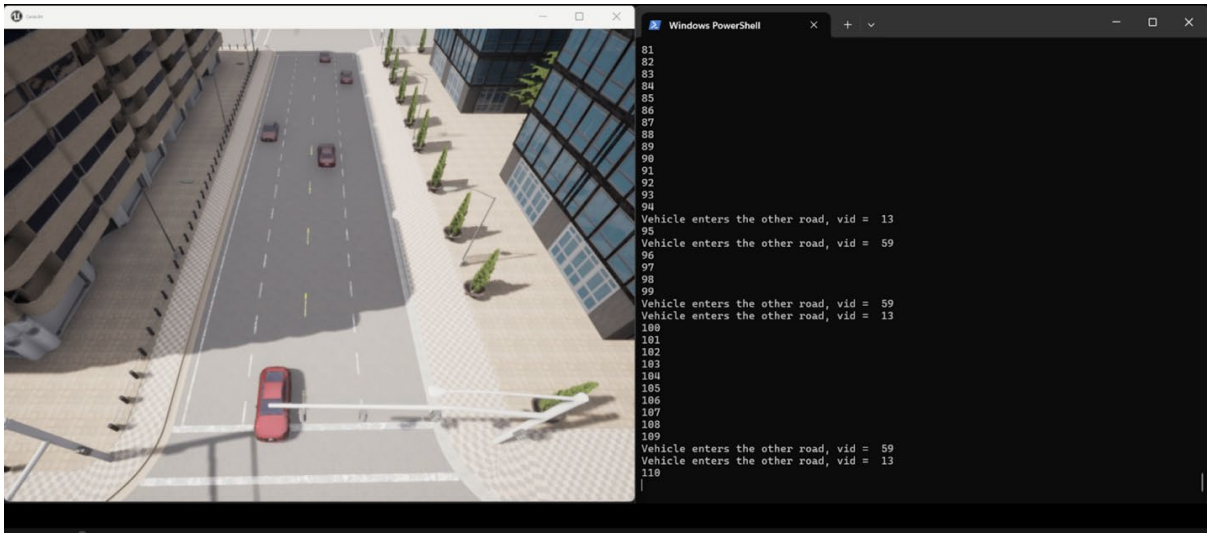

Figure 4: Co-simulation demo using the CARLA Town05 map

6.3 Integration with Scenic

We explore the possibility of combining the simulation with Scenic(Fremont et al. 2019) to enable the systematic generation of high-level, structured scenarios and to interface with diverse autonomous vehicle control stacks. This integration aims to enhance the robustness testing of advanced CAV systems and support the design of more complex and realistic attack scenarios.

Our proposed architecture introduces a co-simulation framework that distributes simulation responsibilities across multiple platforms, leveraging each system's strengths. To mitigate communication overhead, we introduce the high-fidelity bubble concept: vehicles within a defined subregion (the "bubble") are simulated in high fidelity and synchronized at a high frequency, while those outside the bubble are managed by a high-level simulator like METS-R SIM with lower update rates.

Implementing this architecture involves several key components:

- Updating co-simulation region definitions in METS-R SIM
- Enabling dynamic road-level simulation control
- Providing new APIs to interface with Scenic
- Supporting synchronization and region updates on the Scenic side

6.4 Autoware in CARLA

As part of the Scenic integration effort, we also established a communication bridge between CARLA and Autoware[4], an open-source autonomous driving platform. This bridge enables full-stack execution within CARLA's simulated environment. Due to ongoing work on dependencies,

[4] https://github.com/autowarefoundation/autoware

the components are currently maintained within a forked version of VerifAI[5], with installation improvements underway for better usability.

A demonstration of this integration has been recorded[6], and Figure 5 shows a screenshot of Autoware operating within CARLA using this setup.

Figure 5: Autoware operation in CARLA

6.5 Integration of Autoware with Scenic

To support full-stack autonomous driving software (e.g., Autoware) within a multi-simulator environment, we developed a connection manager that coordinates execution among Scenic, CARLA, and the Autoware controller. This integration ensures proper synchronization and message exchange across modules, enabling more advanced testing workflows.

Because Autoware was originally designed for deployment on real vehicles—not simulated platforms, this integration required non-trivial adaptation. The connection manager handles simulation timing and communication protocols to ensure that scenario logic, vehicle dynamics, and control signals remain consistent throughout execution.

Despite minor inconsistencies in scenario sampling, we successfully conducted preliminary experiments involving multiple autonomous vehicles. These results validate the feasibility of running full-stack autonomy in simulation. Future work will focus on improving execution efficiency to support larger-scale testing and faster iteration.

[5] https://github.com/BerkeleyLearnVerify/VerifAI
[6] https://drive.google.com/file/d/1xyjN3UgY0vCBJ6l-_cly8Q90Lt4cl_5m/view

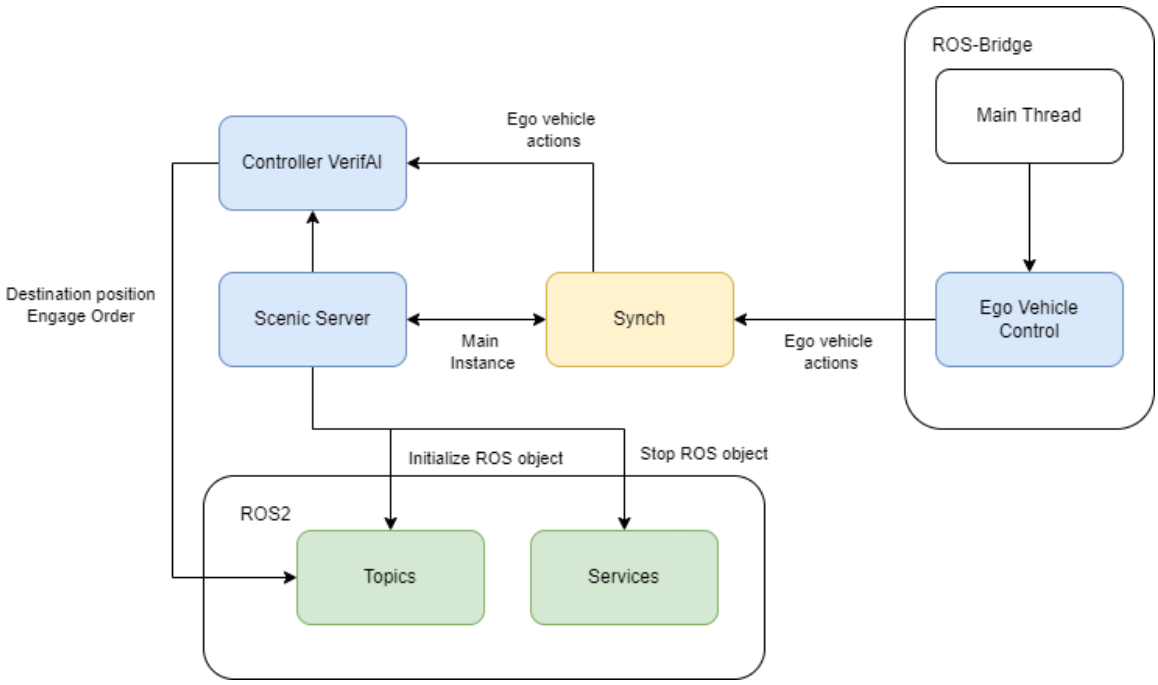

Figure 6: Current Scenic Bridge implementation

CHAPTER 7

System Validation and Platform Refinement

We recognize that the effectiveness and realism of our simulation framework depend heavily on the configuration of scenario parameters. To support flexible experimentation and iterative development, we implement the attacks in interactive notebooks[7] that allow users to customize attack scenarios, control simulation parameters, and visualize system behavior under various conditions. The key characteristics, implementation details, and evaluation settings of each attack are summarized in Table 7.

Table 7: Summary of case studies

CASE STUDY	EXPERIMENTAL SETTING	QUANTITATIVE METRICS
ADVERSARIAL BRAKING	Six-vehicle platoon; additional experiment with 100 background vehicles	Minimum headway, minimum TTC, peak deceleration, travel delay, percentage reduction in completed trips
BSM REPLAY/SPOOFING	One target CAV followed by nine human-driven vehicles	Minimum speed, peak deceleration, speed-recovery time, queue delay, disturbance propagation distance
ROUTE GUIDANCE ATTACK	200 airport-to-Manhattan trips; link weight multiplied by 10,000; random detour variant	Mean and 95th-percentile travel time, route distance, arrival rate, affected links, statistical variation
ADVERSARIAL BOOKING	30 fake requests every 50 ticks for 60 cycles	Zone-level vehicle imbalance, empty vehicle miles, passenger wait time, service rate, unmatched demand
DOS ATTACK	Up to 1,000 packets per second	Median and 95th/99th-percentile latency, packet-loss percentage, CPU utilization, warning detection delay

7.1 Attack identifications for validation

We identify five cyberattacks that can be realistically tested in the simulation testbed:

1. **Adversarial Braking Attack:** A leading vehicle, under attacker control, manipulates its braking pattern to trigger unsafe spacing and potential collisions among autonomous followers, while avoiding collision itself.
2. **Basic Safety Message (BSM) Spoofing:** Attackers exploit V2X channels to inject false safety messages. This may cause vehicles to brake unexpectedly or execute unsafe lane changes. Potential motivations include political activism, ransom-driven disruption, or insider sabotage.

[7] See https://github.com/umnilab/METS-R_HPC/blob/master/tutorials/security_examples.ipynb

3. **Adversarial Booking Attack:** Malicious accounts inject fake ride requests with plausible locations, later cancelling them after a delay (e.g., 3 minutes). This behavior aims to reduce successful matches, misallocate fleet resources, and induce urban congestion.
4. **Route Guidance Attack (RGA):** Through GPS spoofing, attackers mislead vehicles into inefficient or congested routes. An analytical framework based on the Generalized Bathtub Model and an RGA simulation engine quantify rerouting effects and traffic degradation.
5. **Denial-of-Service (DoS) Attack:** An attacker floods the C-V2X communication channel (e.g., UDP port 9001) with high-frequency, syntactically valid Basic Safety Messages (BSMs), overwhelming vehicle or roadside receivers. This disrupts the timely delivery of legitimate safety messages, leading to increased latency, packet loss, and potential failure of real-time safety-critical applications.

7.2 Adversarial braking attack

To evaluate the vulnerabilities of adaptive cruise control (ACC) systems to adversarial driving behaviors, we conduct a series of simulations involving a longitudinal fleet of N vehicles. Each vehicle operates under a switching control policy that alternates between two modes: Cruise Control (CC) and Adaptive Cruise Control (ACC). In CC mode, vehicles use a proportional-integral-derivative (PID) controller to maintain a target speed. In ACC mode, a linear feedback controller is used to preserve a safe following distance from the vehicle ahead.

7.2.1 Threat model

We adopt a threat model inspired by the formal definition introduced in the D4+ framework (Barbosa et al. 2025), which uses Metric Temporal Logic (MTL) to describe adversarial objectives. In our scenario, the attacker controls the lead vehicle (vehicle 0) and can issue arbitrary throttle and brake commands ranging from -1 (max brake) to 1 (max throttle). The attacker has two objectives:

1. Cause traffic instability between at least one pair of trailing vehicles (vehicles 1 to N).
2. Avoid being rear-ended, i.e., maintain a safe distance from the first follower vehicle.

7.2.2 Implementation

In the full D4+ framework, two types of attacks are considered:

- Parametric Attacks: The attacker uses a pre-defined function (e.g., sinusoidal or intermittent stop-and-go) to generate throttle/brake actions. Optimization algorithms adjust parameters to achieve the attack objective.
- Non-Parametric Attacks: The attacker directly selects control values at fixed intervals (e.g., every 6 seconds), with spline interpolation used to create smooth control inputs between time points.

Both versions use global optimization techniques, such as Bayesian Optimization (BO) and Cross-Entropy (CE), to maximize a robustness violation score derived from MTL specifications.

In our simplified setup, we focus on a scripted (non-optimized) version of the attack. The simulation features a fleet of autonomous vehicles governed by platooning algorithms operating

within a corridor of the CARLA Town06 map. The attacker vehicle is positioned at the front of the platoon and repeatedly performs a simple acceleration–deceleration maneuver. This pattern is sufficient to evaluate how adversarial actions can propagate disturbances both upstream and downstream in the traffic flow.

7.2.3 Results

The simulation results are presented in Figure 7 and 8. In the 6 vehicles setup (Figure 6), the attacker's sudden braking causes abrupt decelerations in trailing vehicles. Time-space diagrams show trajectory compression, i.e., increased risk of collisions, and more time consumed for the same distance.

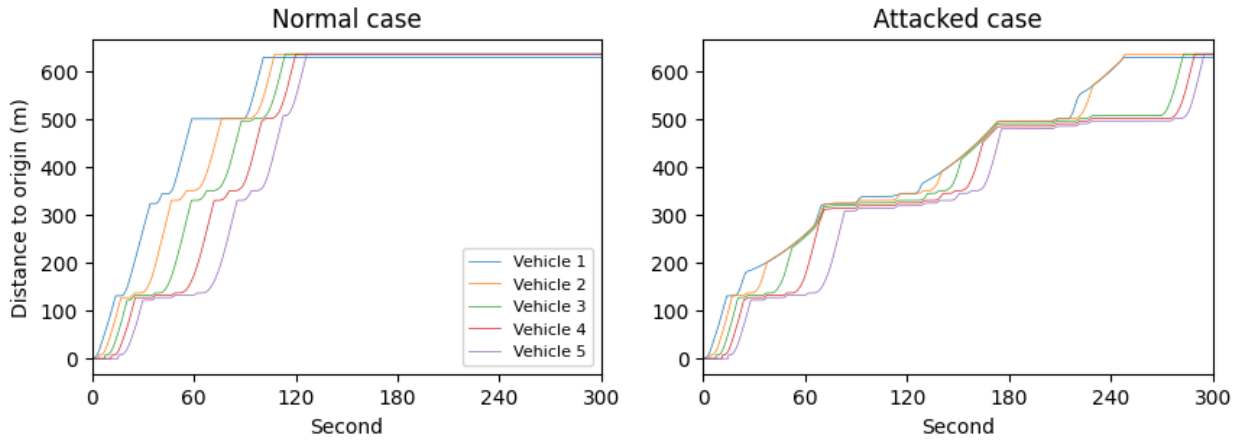

Figure 7: Time-space diagram in attacked and normal cases

In a more complex setting with 100 background vehicles, we observe a decrease in the number of vehicles reaching their destinations (Figure 8). The attack leads to localized congestion propagating to other parts of the networks, demonstrating network-level consequences.

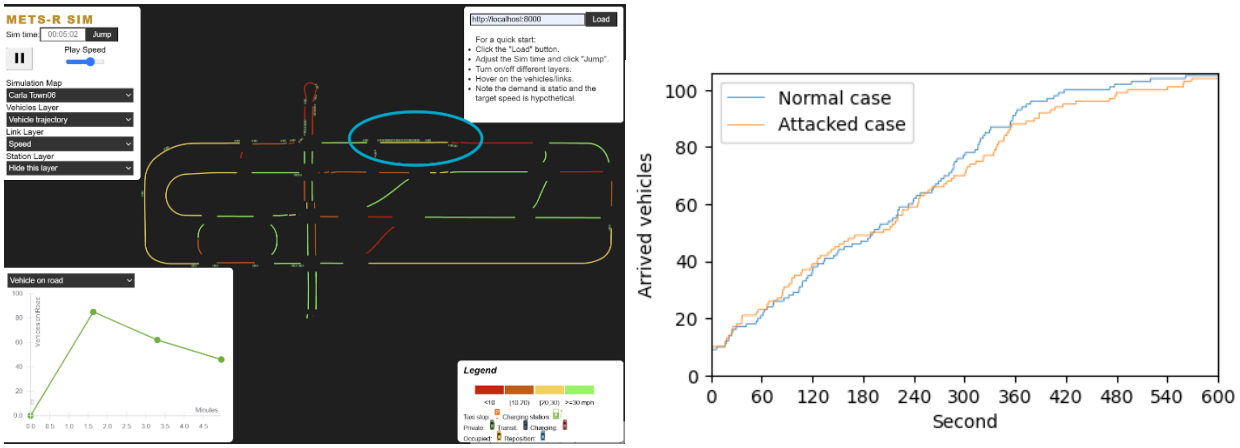

Figure 8: Simulation map (left) and number of arrived vehicles in attacked and normal cases (right). The blue circle highlights the attacked platooning fleet

7.3 Basic safety messages attack

We define a comprehensive threat model for Basic Safety Message (BSM) attacks, which target the vehicle decision-making logic by exploiting V2X communication in traffic safety applications. These attacks aim to influence the behavior of connected and autonomous vehicles (CAVs) by injecting falsified or manipulated data streams, such as Basic Safety Messages (BSMs), into trusted communication channels.

7.3.1 Threat model

The DSM tampering attack assumes an adversary capable of injecting falsified BSMs into the V2X communication network. These messages may mimic emergency events—such as sudden deceleration, road obstacles, or pedestrian presence—that are intended to trigger defensive maneuvers in nearby vehicles. The attacker may be physically present or remote, with the ability to send messages that adhere to accepted communication standards, making it difficult to distinguish from genuine transmissions.

The threat model encompasses attackers with both general-purpose access to V2X systems and targeted knowledge of the road environment. This combination allows them to construct attacks that are not only technically viable but also contextually plausible, thereby increasing their potential effectiveness. The adversary's actions are opportunistically timed to maximize disruption while avoiding direct detection, often leveraging AI-assisted prediction of traffic flows or vehicle behavior.

7.3.2 Implementation

We evaluate a simplified version of the DSM tampering attack within a co-simulation environment integrating CARLA and METS-R SIM. The scenario is deployed in CARLA's Town07 map and features a connected vehicle (CV) that serves as the target. In the first phase of the attack, the adversary records authentic BSMs corresponding to emergency braking or hazard alerts during normal operation. These messages are then replayed when the CV later enters the same road segment.

The attack exploits the CV's reliance on V2X Information by injecting previously valid but currently false messages into the DSRC channel. Although the physical environment remains unchanged, the CV interprets the spoofed data as indicative of an imminent threat and responds with abrupt braking or evasive maneuvers. No physical obstacles are introduced, making the attack difficult to detect using conventional sensing alone.

This testbed does not optimize attack timing or message content using learning-based approaches; instead, it demonstrates that even straightforward replay-based attacks can be sufficient to induce significant behavioral changes in automated vehicles.

7.3.3 Results

Simulation results confirm that the DSM tampering attack can lead to meaningful disruptions in vehicle dynamics and traffic stability. Under normal conditions, the target vehicle accelerates smoothly and maintains consistent speed. However, upon receiving spoofed BSMs, the same vehicle exhibits abrupt deceleration and oscillatory speed behavior. These effects are illustrated in Figures 9 and 10, which compare vehicle acceleration and velocity over time in both the baseline and attacked scenarios.

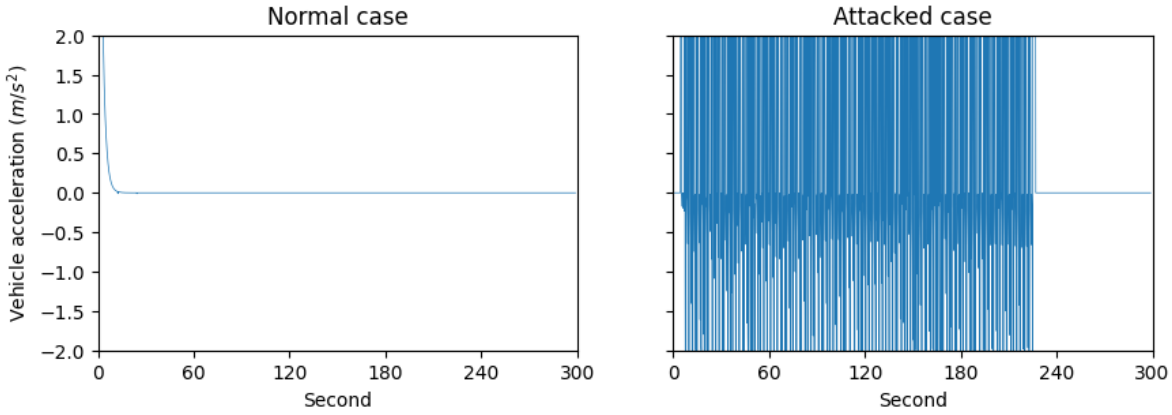

Figure 9: The victim vehicle's acceleration in attacked and normal cases

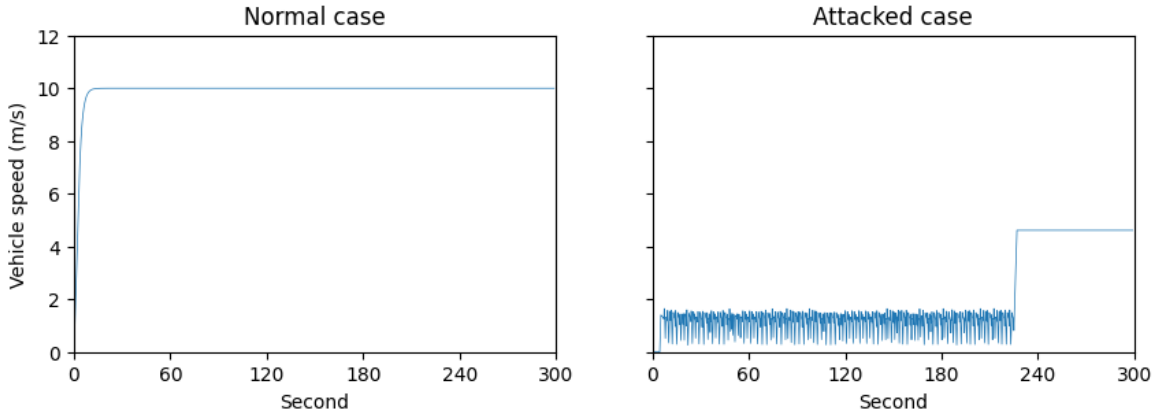

Figure 10: The victim vehicle's speed in attacked and normal cases

To further assess the broader impact, we place 9 additional human-driven vehicles behind the CV to simulate a mixed traffic flow. As shown in Figures 11 and 12, the initial disruption causes a ripple effect throughout the vehicle queue. Trailing vehicles respond to the unexpected

deceleration of the CV, leading to secondary slowdowns and compounding the instability initiated by the attacker.

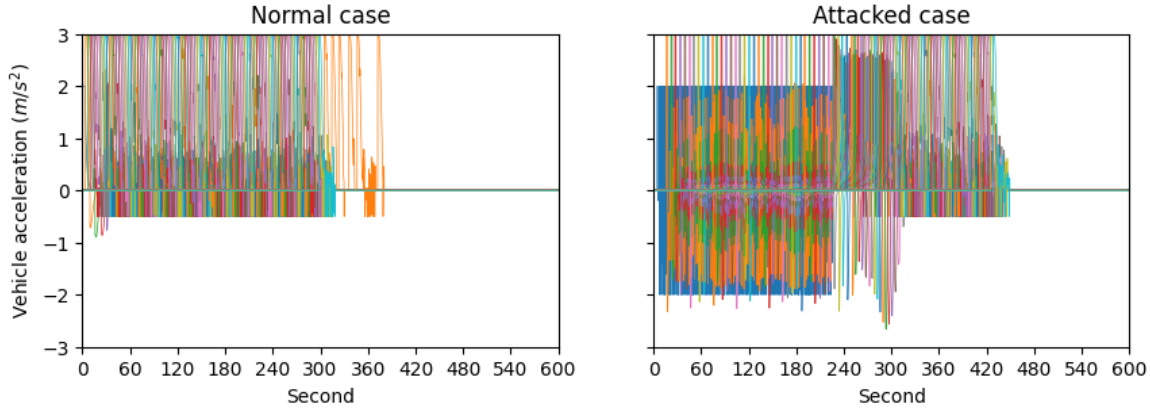

Figure 11: Vehicle accelerations in attacked and normal cases

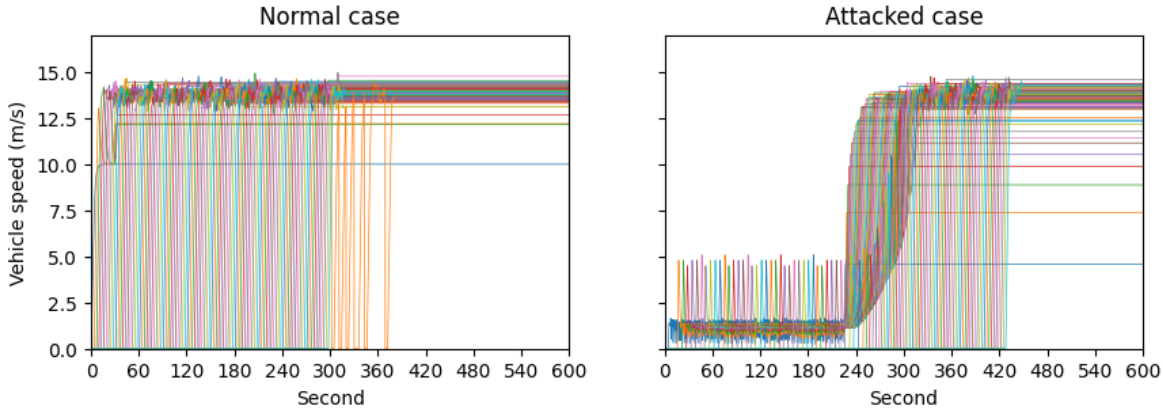

Figure 12: Vehicle speeds in attacked and normal cases

These results highlight how even basic DSM tampering attack, without physical contact, can distort the reliability of V2X-enabled decision-making. They reinforce the need for authenticated message exchange, robust anomaly detection, and cross-validation between perception systems and communication data to ensure the safety and resilience of connected vehicle ecosystems.

7.4 Route guidance attack

Modern navigation systems in connected and autonomous vehicles (CAVs) rely heavily on GPS data and real-time traffic information to determine optimal routing. However, this dependency introduces vulnerabilities that adversaries can exploit through Route Guidance Attacks (RGAs). These attacks involve tampering with location-based services or traffic data to misguide vehicles, thereby increasing congestion, degrading system performance, and even causing safety hazards.

7.4.1 Threat model

We consider a threat model in which the adversary has access to the vehicle's routing logic, either by compromising the onboard navigation system or manipulating external data sources such as V2X messages or GPS signals. The attacker is assumed to possess knowledge of the road network and can inject falsified information that complies with expected formats, making the attack difficult to detect. The goal is to redirect vehicles away from their intended paths by creating the illusion of congestion or road closures, or by modifying cost functions in the routing algorithm to influence path selection.

The model encompasses several adversarial strategies. In a **Weight Manipulation Attack**, the attacker inflates or deflates the cost of specific road segments in the routing graph, thereby making a segment appear highly congested or unusually fast. This is achieved by querying the segment's weight and multiplying it by a malicious factor before updating it in the routing engine. Alternatively, in a **Random Detour Attack**, the adversary forces vehicles to traverse intermediate nodes not originally on their path by GPS spoofing (Zeng et al., 2018), thus extending the route arbitrarily. Both strategies can be deployed in targeted or randomized fashions to affect individuals or entire fleets.

7.4.2 Implementation

The Route Guidance Attack (RGA) is implemented directly within the METS-R SIM simulation framework, which allows fine-grained control over routing logic and network parameters in real time. Two distinct attack strategies are evaluated: the **Weight Manipulation Attack** and the **Random Detour Attack**.

In the **Weight Manipulation Attack**, the adversary alters the perceived travel cost of a specific road segment to mislead the route planning logic. This is achieved by first querying the current weight of the targeted segment and then updating it by applying a multiplicative factor. In the experimental example, we increase the weight by a factor of 10,000 to make the major bridges (Williamsburg Bridge, Queens-Midtown Tunnel, and Ed Koch Queensboro Bridge) in New York City (NYC) appear more congested, discouraging vehicles from selecting them.

The second strategy, the **Random Detour Attack**, forces vehicles to reroute through randomly chosen intermediate nodes, thereby extending their travel paths unnecessarily. In this attack, the vehicle is initially assigned a trip from a designated original road to a destination road. The system retrieves all valid network nodes, excludes the endpoints, and selects an intermediate node at random. Two sub-routes are then queried with one from the origin to the intermediate node, and another from the intermediate node to the destination. These are concatenated into a single detour path and applied to the vehicle's routing logic just before the next simulation tick.

7.4.3 Results

In the weight manipulation scenario, we simulate 200 vehicle trips departing from John F. Kennedy International Airport to downtown Manhattan. The total number of vehicles arriving remains high in both baseline and attack conditions, as shown in Figure 13. This suggests that even

though the attacker could modify link weights on critical segments, such as major bridges in the NYC network, would not lead to significant traffic delays.

For the random detour attack, experiments conducted on the NYC road network reveal clear differences between original and spoofed trajectories, as illustrated in Figure 14.

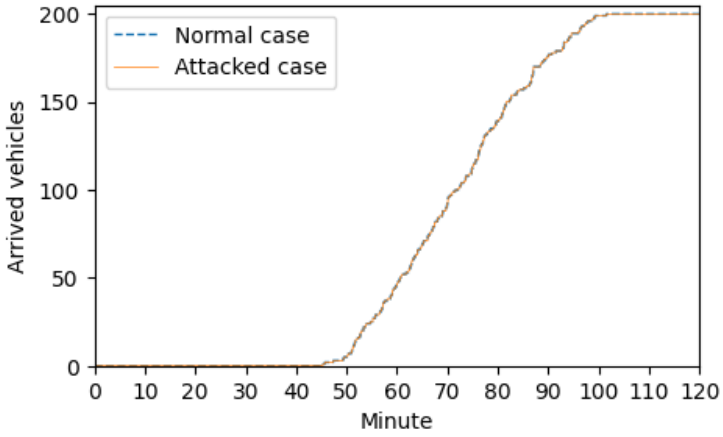

Figure 13: Number of vehicles arrived under Weight Manipulation Attack

Figure 14: Vehicle routes under Random Detour Attack

These findings validate the importance of robust defenses against GPS spoofing and route manipulation, including secure V2X protocols, anomaly detection systems, and resilient routing algorithms. The RGA framework offers a flexible and scalable testbed for evaluating such countermeasures under realistic urban traffic conditions.

7.5 Adversarial booking attack

The adversarial booking attack targets ride-hailing platforms by exploiting vulnerabilities in vehicle-passenger matching algorithms. Specifically, attackers aim to manipulate the spatial distribution of vehicles and reduce system efficiency by injecting fake trip requests into the

platform. These attacks can degrade match quality, induce vehicle congestion in target regions, and increase wait times for legitimate users.
We formalize this problem as a bilevel optimization, where the attacker controls K fake accounts and strategically generates requests to disrupt vehicle allocation decisions made by the platform.

7.5.1 Threat model

In this threat model, we assume an attacker has access to multiple fake or compromised user accounts that can submit trip requests to the ride-hailing system. These accounts may be created using disposable email addresses or phone numbers, or obtained through coordinated efforts (e.g., social engineering, account trading). Each account has the appearance of a legitimate user, making it difficult for the platform to detect anomalies using conventional behavioral signals.

The attacker does not interfere with the platform's internal algorithms or vehicle routing decisions directly. Instead, they exploit the system's reactive nature: vehicles are dispatched or repositioned based on perceived passenger demand. By introducing fake trip requests—particularly in high volumes or strategically selected locations—the attacker can distort supply-demand balance, pull idle vehicles into unproductive zones, or block real passengers from timely service.

More advanced attackers may also model the system response or learn from historical behavior, optimizing when and where to inject requests. In the dynamic extension of this model, the attacker balances long-term goals (e.g., sustained inefficiency) with platform-imposed constraints such as request rate limits, cancellation penalties, or spatial/temporal booking limits. This multi-step attack policy may be enhanced with reinforcement learning (e.g., PPO), allowing the attacker to adapt over time while evading detection.
Overall, the threat model captures a realistic yet potent adversarial capability—one that operates under limited system knowledge, uses only observable actions, and relies on plausible user behavior to degrade service quality in a scalable and persistent manner.

7.5.2 Implementation

We implement a simplified version of the adversarial booking attack using the METS-R SIM framework to assess its real-time impact on fleet dynamics. Rather than solving the full MILP or using an RL-based controller, we demonstrate the effectiveness of repeated fixed-pattern fake bookings over time.

In our test setup, the attacker continuously injects fake trip requests from a fixed pickup zone over a 10-minute simulation horizon. Specifically, the simulation generates 30 fake bookings every 50 ticks, repeated for 60 cycles.

This static and non-adaptive strategy is sufficient to induce measurable shifts in vehicle distribution, highlighting the vulnerability of dispatch systems to unverified requests. The fake demand causes idle vehicles to congregate near the spoofed pickup zone, reducing availability elsewhere and degrading match quality for real passengers. While this approach lacks temporal or spatial optimization, it serves as a practical demonstration of how simple attack strategies can generate disproportionate effects in large-scale mobility systems.

7.5.3 Results

The simulation shows that sustained injection of fake trip requests can significantly distort vehicle distribution in the network. As visualized in the heatmaps (Figure 15), vehicles under normal conditions remain relatively evenly dispersed, while under adversarial booking attack, they cluster densely around the path to the spoofed origin zone. This artificial concentration reduces availability in other regions and demonstrates how even a simple, non-optimized strategy can induce spatial inefficiencies in ride-hailing systems.

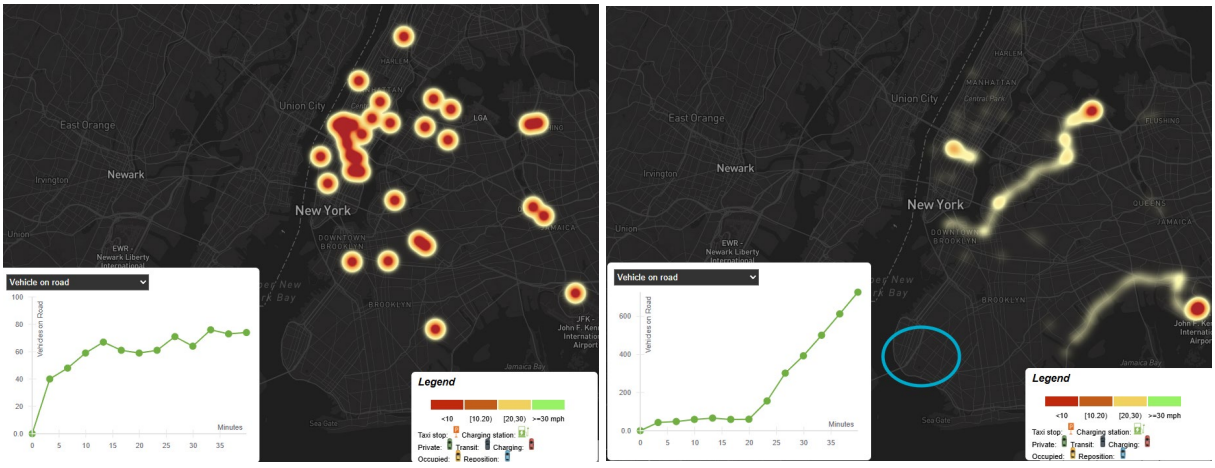

Figure 15: Repositioning vehicle distribution under normal (left) and attacked (right) cases. The blue circle marks the spoofed pickup zone

7.6 Denial-of-Service (DoS) attack

Cellular Vehicle-to-Everything (C-V2X) communication, standardized by the 3rd Generation Partnership Project (3GPP) under Release 14 and beyond, enables direct communication between vehicles, infrastructure, and pedestrians without relying on cellular network coverage. This direct mode, also known as PC5 sidelink, supports low-latency transmission of safety-critical messages such as Basic Safety Messages (BSMs) using the 5.9 GHz spectrum. While C-V2X offers enhanced range and reliability compared to legacy Dedicated Short Range Communication (DSRC)[8], it remains vulnerable to various cyber-physical threats, including Denial-of-Service (DoS) attacks. These vulnerabilities are particularly concerning in the context of real-time safety applications. We investigate the vulnerability of a C-V2X receiver to DoS attacks by simulating high-rate UDP traffic targeting the BSM communication channel, assessing the system's resilience and impact under adversarial load.

7.6.1 Threat model

In this threat scenario, an adversary aims to disrupt the reliability and performance of a C-V2X communication system by launching a DoS attack. The attacker's goal is to:

[8] See https://www.ettifos.com/post/dsrc-vs-cv2x

1. Saturate the communication channel (UDP port 9001) used for transmitting Basic Safety Messages (BSMs).
2. Degrade or delay the delivery of legitimate V2X safety-critical messages.
3. Cause service-level degradation in Roadside Units (RSUs) or On-Board Units (OBUs).

Assumptions:

1. The attacker has access to the same network as the CV2X devices (e.g., via a compromised vehicle or rogue device).
2. The attacker can craft and send valid BSMs or generic UDP packets at a high frequency.
3. C-V2X receiver does not implement advanced intrusion detection or rate-limiting.

7.6.2 Implementation

Accurately simulating communication-layer latency requires full integration with tools such as NS-3, a capability we have explored but not yet fully implemented. To address this limitation, we instead deploy a dedicated physical testbed using pure Python scripts and real C-V2X hardware. This setup enables high-frequency message injection and low-level network manipulation that would be difficult to reproduce in a virtualized environment.

Experimental Setup

1. Attacker Node (192.168.94.100): A Python script is used to flood the target with valid BSMs at configurable high rates (up to 1000 packets/sec).
2. Victim RSU (192.168.94.94): A receiver application listens on UDP port 9001, parses and logs BSMs, and monitors latency and packet rate.
3. Testbed: Communication is conducted over a local area network simulating the C-V2X broadcast domain using Cohda or similar C-V2X-capable hardware.

Tools Used

1. Custom Python BSM sender (udp_flood_sender.py)
2. Custom receiver logger (receiver_logger_with_pps.py)
3. tcpdump for packet trace validation
4. nping and hping3 for alternative flooding tests

Attack Strategy

1. The attacker sends syntactically valid JSON-formatted BSMs at a rate exceeding normal operational levels (e.g., >100 packets/sec).
2. The messages include all required BSM fields to bypass basic input validation.
3. The receiver logs each packet:
    a. Arrival time
    b. Latency (based on embedded timestamp)
    c. Validity
    d. Packet-per-second rate

7.6.3 Results

Observed Metrics

- Latency Spikes: As the packet rate increased beyond 500 PPS, the receiver began showing increased latency due to processing delays.
- Throughput Saturation: At 1000 PPS, packet drops were observed. The receiver could no longer log every packet, indicating buffer overflow.
- PPS Detection: The enhanced receiver successfully triggered DoS warnings when packet-per-second exceeded configured thresholds (e.g., 500 PPS).

System Impact:

- Logging delays and occasional CPU saturation were observed on the RSU.
- Safety alert notification was delayed during flooding.
- Time-series plots of latency showed sharp increases during attack windows.
- PPS graphs confirmed the flooding behavior and its impact.

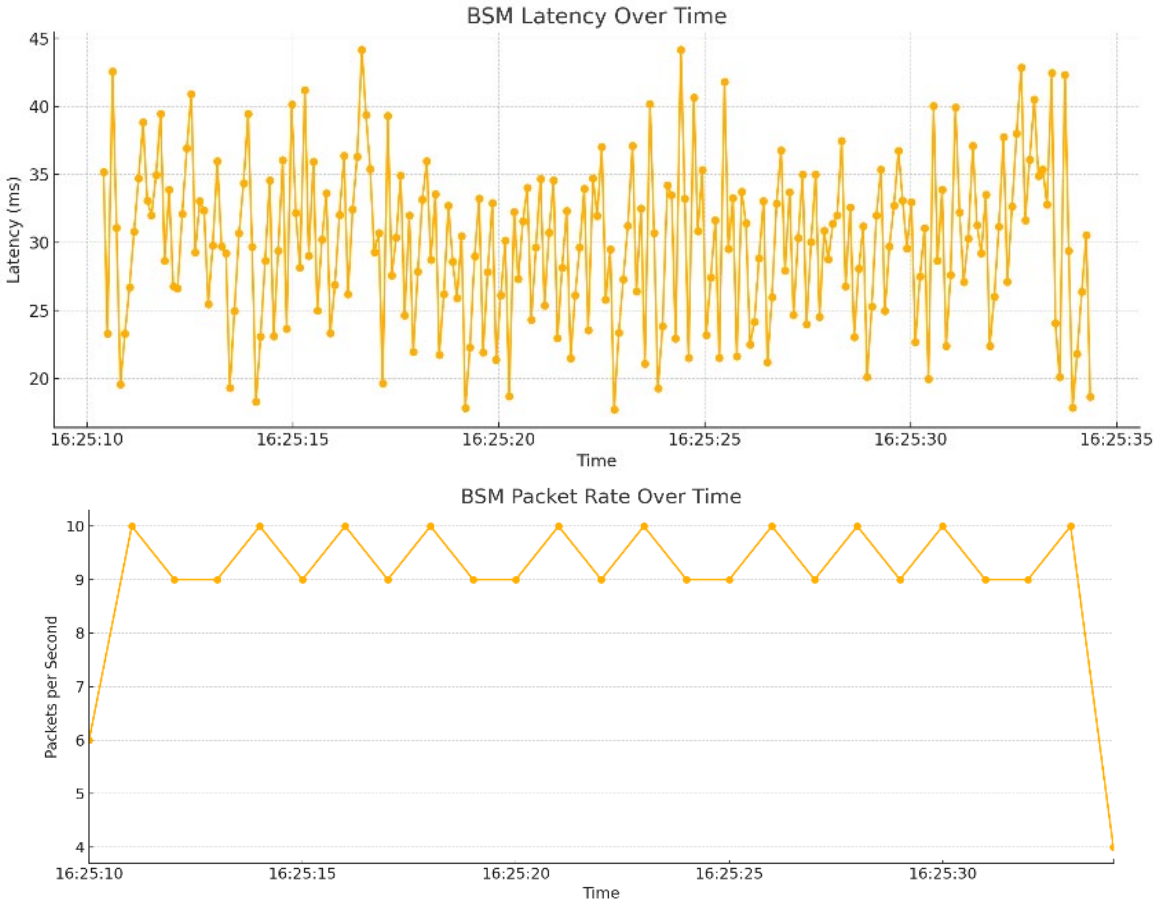

Figure 16: Time-series plots of latency and BSM packet rate before attack

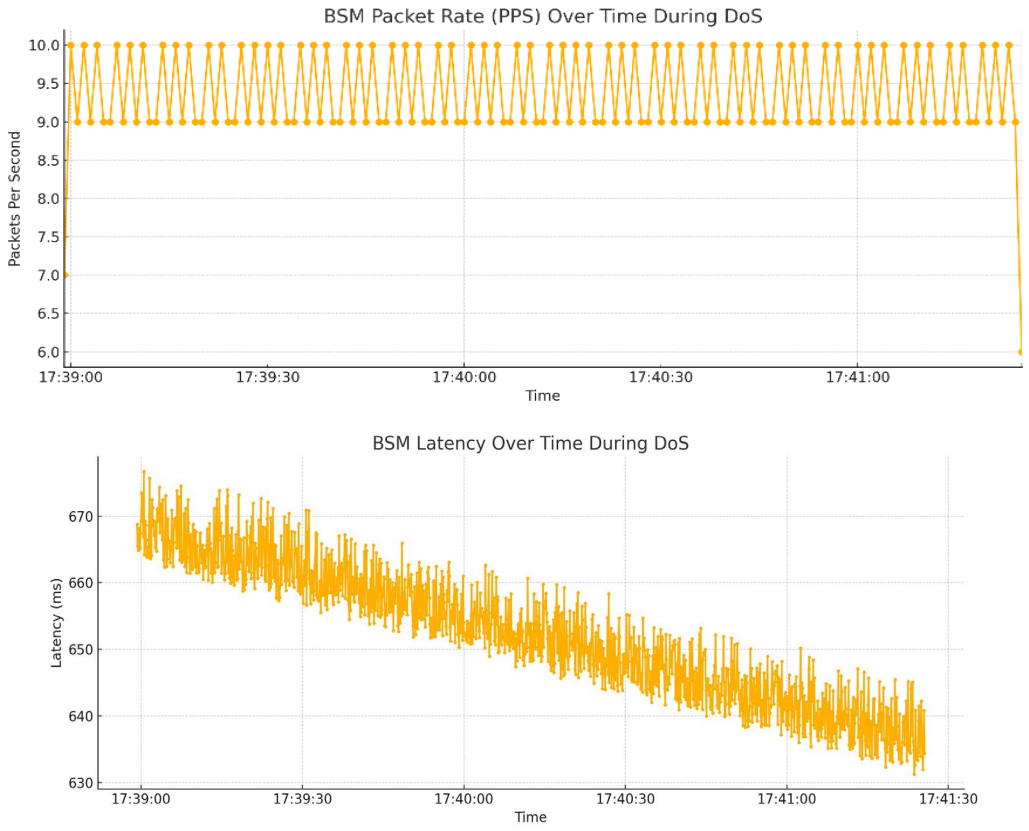

Figure 17: Time-series plots of latency and BSM packet rate after attack

CHAPTER 8

Conclusions and Discussion

8.1 Project achievement

This project developed a comprehensive simulation testbed to evaluate the cybersecurity vulnerabilities of Connected and Autonomous Vehicle (CAV) systems. The platform integrates several core components: **METS-R SIM** for large-scale traffic modeling, **METS-R HPC** for synchronizing simulation components and exposing Step, Query, and Control APIs for external monitoring and intervention, **CARLA** for vehicle-level dynamics and sensor realism, and **Kafka** for an application-level data-streaming layer for BSMs, link travel times, and link energy consumption, allowing attacks and defenses to interact with transportation data in real time. Together, these modules enable detailed examination of attack impacts across spatial and functional layers of transportation systems.

The platform was validated using five classes of cyberattacks spanning vehicle control, V2X messaging, route guidance, fleet operations, and communication services. In the Town05 co-simulation demonstration, the testbed generated approximately 100 trips per simulated minute, representing 1,000 trips over a 10-minute simulation horizon. The co-simulation interface was also extended to support eight official CARLA maps. In the physical C-V2X denial-of-service experiment, receiver latency began to increase when malicious traffic exceeded approximately 500 packets per second, and packet loss was observed at 1,000 packets per second. The receiver also correctly triggered threshold-based warnings when the packet rate exceeded the configured limit of 500 packets per second.

The individual case studies further demonstrated the platform's ability to represent attacks at meaningful operational scales. The adversarial-braking experiments included both a 6-vehicle platoon and a larger scenario with 100 background vehicles. The replayed BSM attack involved one target vehicle and nine additional human-driven vehicles. The route-guidance attack was evaluated using 200 trips and a 10,000-fold modification of the affected link weight, while the fraudulent-booking attack injected 30 requests per cycle over 60 cycles. Collectively, these case studies provided proof-of-concept evidence that cyberattacks can propagate across system layers and produce network-level consequences. Adversarial braking disrupted traffic progression beyond the directly attacked platoon; replayed safety messages induced oscillatory behavior in the target vehicle and secondary disturbances among the 9 following vehicles; route manipulation altered vehicle trajectories; and repeated fake requests caused idle vehicles to accumulate near the spoofed pickup location.

8.2 Project outputs and technology transfer

The project translated its technical developments into publicly accessible and reusable research artifacts. The primary software outputs include METS-R SIM for network-scale transportation simulation and METS-R HPC for remote orchestration, interactive control, and attack implementation. An interactive security notebook packages representative attacks in a form that allows users to modify scenario parameters, execute experiments, and visualize system responses.

The project also provides a demonstration of Autoware operating in CARLA through the developed communication bridge.

The released artifacts support several technology-transfer pathways. Researchers can use the APIs and notebooks to reproduce the reported attacks, integrate new attack or defense algorithms, and compare outcomes across vehicle-, communication-, traffic-, and fleet-level metrics. Autonomous-driving developers can connect external control stacks to CARLA and evaluate their responses within network-scale traffic conditions. Transportation agencies and cybersecurity practitioners can adapt the scenario configurations for training, tabletop exercises, and preliminary risk assessment. The modular design also supports classroom instruction by separating the traffic, communication, vehicle-control, and attack components into independently configurable modules.

8.3 Limitations and future directions

Although the virtual testbed supports a wide range of simulations, several limitations remain that motivate future development:

- **Bridging Virtual and Physical Testing:** Purely simulated environments may omit certain real-world nuances in communication delays, hardware constraints, or environmental conditions. To address this, we conducted a preliminary Denial-of-Service (DoS) experiment at the Clemson University CAV testbed using three C-V2X devices: one acting as a mobile On-Board Unit (OBU), another as a Roadside Unit (RSU), and a third as the attacker. By flooding the RSU with Basic Safety Messages (BSMs), the attacker was able to overload its processing queue, causing significant latency and disruption in safety-critical communications. This confirmed the feasibility of real-world DoS attacks on connected vehicle applications.
- **Hybrid Simulation and Hardware Integration:** In the next phase, we will integrate physical hardware and field data into the simulation loop. This includes linking real C-V2X units to the Kafka data stream, capturing actual sensor data for anomaly detection, and testing real-time defensive responses through hardware-in-the-loop simulations.
- **Scenario Realism and Automation:** While the testbed supports manual scenario configuration, realistic threat scenarios require automated synthesis based on real-world traffic patterns, behavioral models, and infrastructure constraints. We are developing interactive tools and notebooks for scenario refinement, though further work is needed to incorporate adaptive and data-driven threat modeling.

REFERENCES

Barbosa, D.O., Burbano, L., Hernandez, C., Lei, Z., Park, Y., Ukkusuri, S. and Cardenas, A.A., 2025. D4+: Emergent Adversarial Driving Maneuvers with Approximate Functional Optimization. *arXiv preprint arXiv:2505.13942*.

Cantas, M.R. and Guvenc, L., 2023. Customized co-simulation environment for autonomous driving algorithm development and evaluation. arXiv preprint arXiv:2306.00223.

Chen, Q.A., Yin, Y., Feng, Y., Mao, Z.M. and Liu, H.X., 2018, February. Exposing Congestion Attack on Emerging Connected Vehicle based Traffic Signal Control. In NDSS.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A. and Koltun, V., 2017, October. CARLA: An open urban driving simulator. In Conference on robot learning (pp. 1–16). PMLR.

Fayyad, J., Jaradat, M.A., Gruyer, D. and Najjaran, H., 2020. Deep learning sensor fusion for autonomous vehicle perception and localization: A review. Sensors, 20(15), p.4220.

Fremont, D.J., Dreossi, T., Ghosh, S., Yue, X., Sangiovanni-Vincentelli, A.L. and Seshia, S.A., 2019, June. Scenic: a language for scenario specification and scene generation. In Proceedings of the 40th ACM SIGPLAN conference on programming language design and implementation (pp. 63–78).

Garikapati, D. and Shetiya, S.S., 2024. Autonomous vehicles: Evolution of artificial intelligence and learning algorithms. arXiv preprint arXiv:2402.17690.

Gelbal, S.Y., Zhu, S., Anantharaman, G.A., Guvenc, B.A. and Guvenc, L., 2023. Cooperative collision avoidance in a connected vehicle environment. arXiv preprint arXiv:2306.01889.

HomChaudhuri, B. and Bhattacharyya, V., 2022. Distributed model predictive control for connected and automated vehicles in the presence of uncertainty. Journal of Autonomous Vehicles and Systems, 2(1), p.011004.

Huang, K., Shi, B., Li, X., Li, X., Huang, S. and Li, Y., 2022. Multi-modal sensor fusion for auto driving perception: A survey. arXiv preprint arXiv:2202.02703.

Hussein, A., Díaz-Álvarez, A., Armingol, J.M. and Olaverri-Monreal, C., 2018, November. 3DCoAutoSim: Simulator for cooperative ADAS and automated vehicles. In 2018 21st International Conference on Intelligent Transportation Systems (ITSC) (pp. 3014–3019). IEEE.

Khan, A.R., Jamlos, M.F., Osman, N., Ishak, M.I., Dzaharudin, F., Yeow, Y.K. and Khairi, K.A., 2022. DSRC technology in Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) IoT system for Intelligent Transportation System (ITS): A review. In Recent trends in mechatronics towards industry 4.0, pp.97–106.

Lei, Z., Xue, J., Chen, X., Qian, X., Saumya, C., He, M., Sobolevsky, S., Kulkarni, M. and Ukkusuri, S.V., 2024. METS-R SIM: A simulator for Multi-modal Energy-optimal Trip Scheduling in Real-time with shared autonomous electric vehicles. Simulation Modelling Practice and Theory, 132, p.102898.

Li, J., Chen, C. and Yang, B., 2024. V2X assisted co-design of motion planning and control for connected automated vehicle. IET Intelligent Transport Systems, 18(12), pp.2601–2617.

Li, J., Hong, D., Gao, L., Yao, J., Zheng, K., Zhang, B. and Chanussot, J., 2022. Deep learning in multimodal remote sensing data fusion: A comprehensive review. International Journal of Applied Earth Observation and Geoinformation, 112, p.102926.

Li, P., Kusari, A. and LeBlanc, D.J., 2021a. A novel traffic simulation framework for testing autonomous vehicles using SUMO and CARLA. arXiv preprint arXiv:2110.07111.

Li, S., Shu, K., Chen, C. and Cao, D., 2021b. Planning and decision-making for connected autonomous vehicles at road intersections: A review. Chinese Journal of Mechanical Engineering, 34, pp.1–18.

Lopez, P.A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P. and Wießner, E., 2018, November. Microscopic traffic simulation using SUMO. In 2018 21st international conference on intelligent transportation systems (ITSC) (pp. 2575–2582). IEEE.

Malik, R.Q., Ramli, K.N., Kareem, Z.H., Habelalmatee, M.I., Abbas, A.H. and Alamoody, A., 2020, September. An overview on V2P communication system: Architecture and application. In 2020 3rd International Conference on Engineering Technology and its Applications (IICETA) (pp. 174–178). IEEE.

Ngo, H., Fang, H. and Wang, H., 2023. Cooperative perception with V2V communication for autonomous vehicles. IEEE Transactions on Vehicular Technology, 72(9), pp.11122–11131.

Oroko, J.A. and Nyakoe, G.N., 2022, June. Obstacle avoidance and path planning schemes for autonomous navigation of a mobile robot: a review. In Proceedings of the sustainable research and innovation conference (pp. 314–318).

Parekh, D., Poddar, N., Rajpurkar, A., Chahal, M., Kumar, N., Joshi, G.P. and Cho, W., 2022. A review on autonomous vehicles: Progress, methods and challenges. Electronics, 11(14), p.2162.

Park, G. and Choi, M., 2023. Optimal path planning for autonomous vehicles using artificial potential field algorithm. International Journal of Automotive Technology, 24(5), pp.1259–1267.

Rana, M.M. and Hossain, K., 2023. Connected and autonomous vehicles and infrastructures: A literature review. International Journal of Pavement Research and Technology, 16(2), pp.264–284.

Ribeiro, B., Nicolau, M.J. and Santos, A., 2023. Using machine learning on V2X communications data for VRU collision prediction. Sensors, 23(3), p.1260.

Ross, R.S., 2012. Guide for conducting risk assessments. In NIST Pubs.

Szendrei, Z., Varga, N. and Bokor, L., 2018. A SUMO-based hardware-in-the-loop V2X simulation framework for testing and rapid prototyping of cooperative vehicular applications. In Vehicle and Automotive Engineering 2 (pp. 426–440). Springer.

Vrbanić, F., Čakija, D., Kušić, K. and Ivanjko, E., 2021. Traffic flow simulators with connected and autonomous vehicles: A short review. Transformation of Transportation, pp.15–30.

Yan, C., Xu, Z., Yin, Z., Ji, X. and Xu, W., 2022. Rolling colors: Adversarial laser exploits against traffic light recognition. In 31st USENIX Security Symposium (USENIX Security 22) (pp. 1957–1974).

Yeong, D.J., Velasco-Hernandez, G., Barry, J. and Walsh, J., 2021. Sensor and sensor fusion technology in autonomous vehicles: A review. Sensors, 21(6), p.2140.

Zeng, K.C., Liu, S., Shu, Y., Wang, D., Li, H., Dou, Y., Wang, G. and Yang, Y., 2018. All your GPS are belong to us: Towards stealthy manipulation of road navigation systems. In 27th USENIX security symposium (USENIX security 18) (pp. 1527–1544).

Zhang, J., Chang, C., He, Z., Zhong, W., Yao, D., Li, S. and Li, L., 2023. CAVSim: A microscopic traffic simulator for evaluation of connected and automated vehicles. IEEE Transactions on Intelligent Transportation Systems, 24(9), pp.10038–10054.

Zhuang, Y., Sun, X., Li, Y., Huai, J., Hua, L., Yang, X., Cao, X., Zhang, P., Cao, Y., Qi, L. and Yang, J., 2023. Multi-sensor integrated navigation/positioning systems using data fusion: From analytics-based to learning-based approaches. Information Fusion, 95, pp.62–90.