Project Title : Detecting false data injection attack on vehicle to cloud communication in cloud connected autonomous vehicles.

Detecting false data injection attack on vehicle to cloud communication in cloud connected autonomous vehicles

INTRODUCTION

- Advancements in autonomous vehicle technology have led to increased reliance on cloud-based services for data transmission and processing.

- Cloud-connected autonomous vehicles leverage cloud infrastructure for data storage, processing, and communication, enabling real-time decision-making and enhanced functionality.

- However, the integration of cloud services introduces security vulnerabilities, particularly concerning the integrity of data transmitted from vehicles to the cloud.

PROBLEM STATEMENT

- False data injection can compromise the integrity and safety of autonomous vehicles by tampering with vehicle data, potentially leading to incorrect decisions and unsafe driving conditions.

- Our project aims to detect false data injection attacks on the communication channel between autonomous vehicles and the cloud infrastructure, known as the Vehicle-to-Cloud (V2C) communication channel.

Abstract and Scope

Project Overview:

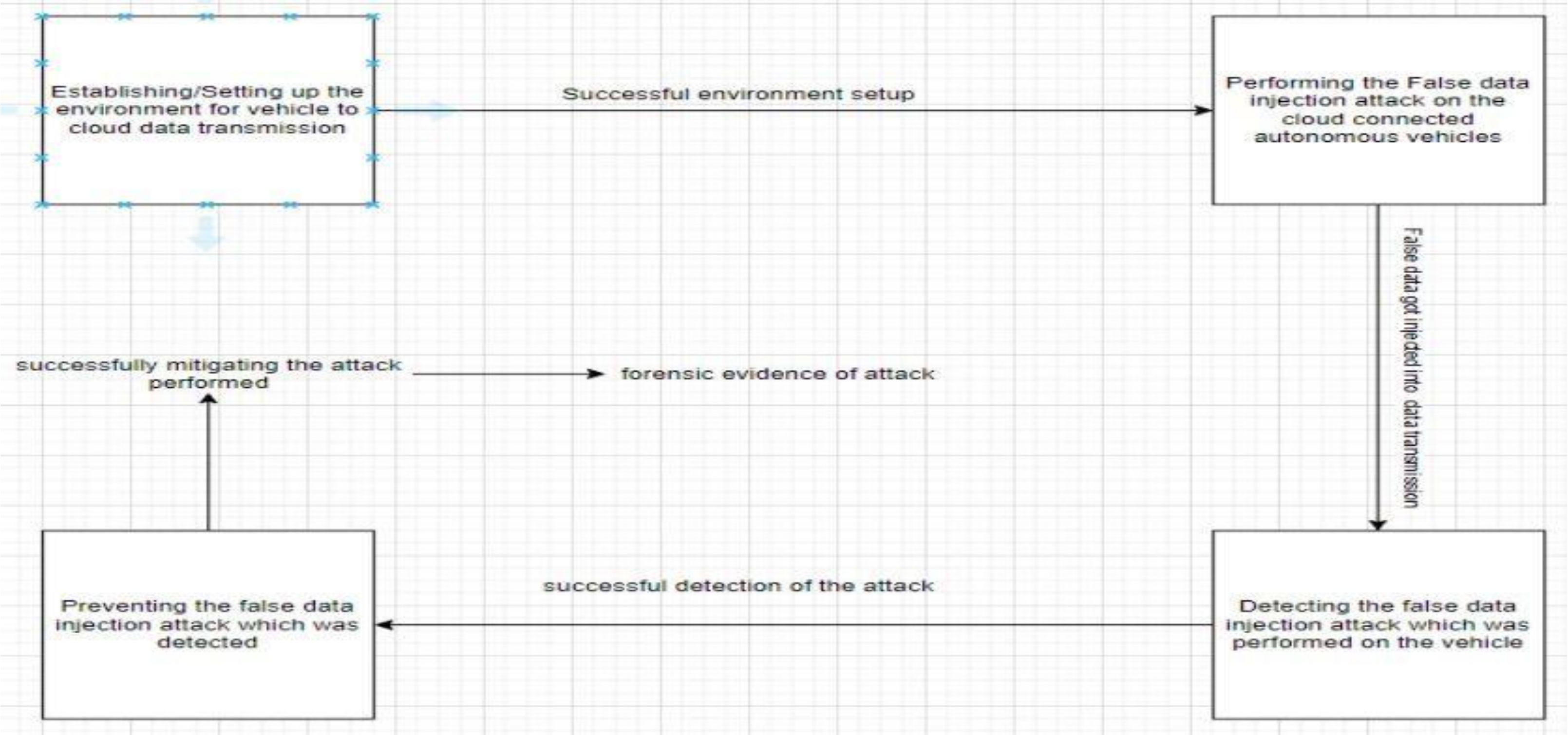
- Our project focuses on detecting false data injection attacks in cloud-connected autonomous vehicles through a comprehensive detection strategy.
- Utilizing data analysis and statistical techniques for anomaly detection.
- Implementing machine learning models for identifying deviations in vehicle data.
- Upon detecting an attack, the system will trigger alert notifications to recommend driver intervention (Level 3 automation) for vehicle control and safety.

Suggestions from Review - 3

Suggestions from Panel Members:

- Show 3d simulation to illustrate the attack scenarios.
- Integrate MobatSim with OpenStreetMap for better visualization.
- Check cloud connectivity feasibility.
- Show and collect all the vehicle data which can be sent from vehicle to cloud.

Design Approach


Establishing/Setting up the environment for vehicle to cloud data transmission
Successful environment setup
Performing the False data injection attack on the cloud connected autonomous vehicles
False data got injected into data transmission
Detecting the false data injection attack which was performed on the vehicle
successful detection of the attack
Preventing the false data injection attack which was detected
successfully mitigating the attack performed
forensic evidence of attack

Design Approach

Here are the steps involved in the design approach:

1. Establishing/Setting up the environment for vehicle to cloud data transmission:

This step involves setting up simulation part for v2c(vehicle to cloud) communication.

2. Performing the False data injection attack on the cloud connected autonomous vehicles: This step depicts the attacker injecting false data into the communication channel.

Design Approach

3. Detecting the false data injection attack:

This step involves monitoring the data transmission for any anomalies that could indicate a false data injection attack.

4. Stopping the false data injection attack: Once a false data injection attack is detected, the system can take steps to prevent it from being successful. This could involve alerting the driver to take control of the vehicle.

Design Approach

Autonomous Vehicle Setup & Cloud Integration

Autonomous Vehicle Simulation with MOBATSim:

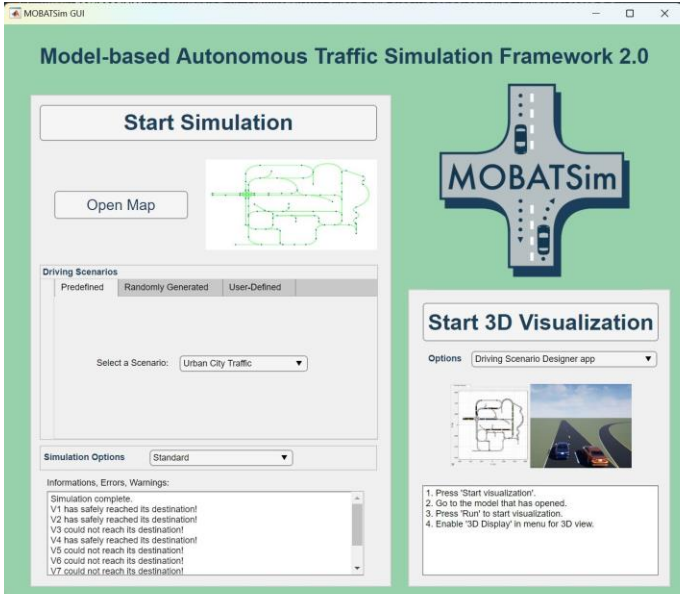
- Using MOBATSim, a MATLAB-based simulation framework for autonomous traffic simulations.
- Customizable models and code for automated driving systems, including path planning and control.
- Simulate vehicle dynamics and sensor data within a controlled environment.
- Compatible with cloud integration

Cloud Integration for Data Transmission:

- Deploying cloud infrastructure on AWS (Amazon Web Services).
- Facilitates real-time vehicle-to-cloud communication.

Design Approach

Vehicle Simulation Front-end



Select a Scenario: Urban City Traffic

Simulation Options: Standard

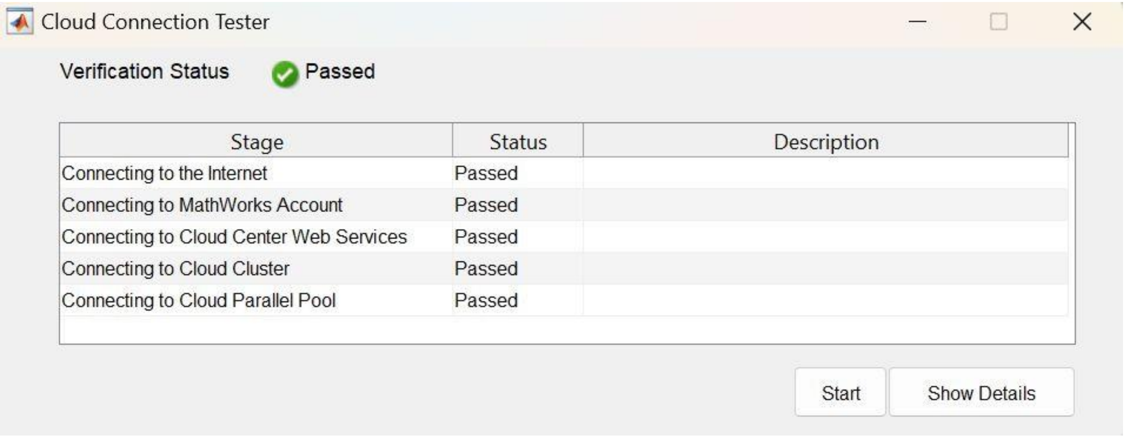
Options: Driving Scenario Designer app

Informations, Errors, Warnings:

```
Simulation complete.
V1 has safely reached its destination!
V2 has safely reached its destination!
V3 could not reach its destination!
V4 has safely reached its destination!
V5 could not reach its destination!
V6 could not reach its destination!
V7 could not reach its destination!
```

1. Press 'Start visualization'.
2. Go to the model that has opened.
3. Press 'Run' to start visualization.
4. Enable '3D Display' in menu for 3D view.

Successfully tested for cloud connectivity

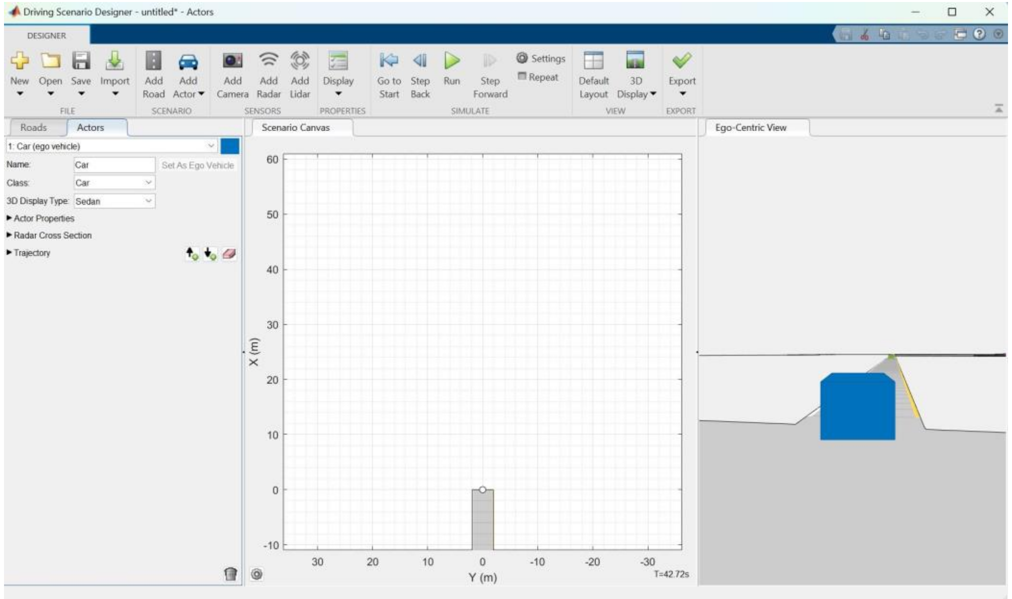


Verification Status ✓ Passed

Stage	Status	Description
Connecting to the Internet	Passed	
Connecting to MathWorks Account	Passed	
Connecting to Cloud Center Web Services	Passed	
Connecting to Cloud Cluster	Passed	
Connecting to Cloud Parallel Pool	Passed	

Design Approach

2D visualization of the path of the vehicle



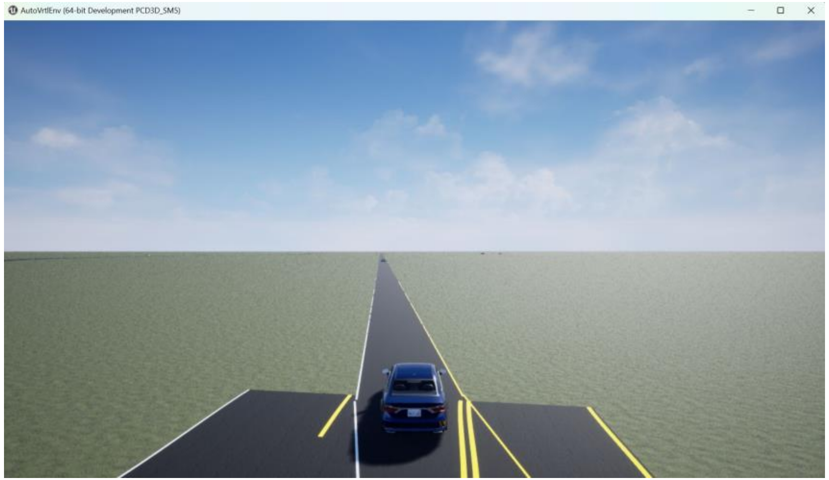
1: Car (ego vehicle)

Name: Car

Class: Car

3D Display Type: Sedan

3D Simulation



Cloud Architecture

a. Frontend: The frontend layer serves as the interface for users and vehicles to interact with the cloud system. It serves as the web interface for monitoring vehicle data.

b. Backend: The backend layer manages the core functionality of the system. It includes several components:

i. API Gateway: Acts as a single entry point for all communication between the vehicle simulation and the cloud services. It routes requests to appropriate microservices and handles authentication.

ii. Microservices: Modular components responsible for specific tasks such as data ingestion, validation, processing, and storage. Each microservice is designed to handle a specific function.

Cloud Architecture

iii. Database: A scalable and reliable database system will be used to store vehicle data. We will use a relational database as it is capable of storing time-series data efficiently.

iv. Security Layer: This is where we implement our security measures, i.e, an intrusion detection system to safeguard against false data injection attacks.

v. Monitoring and Logging: Tools for monitoring system health, performance, and logging activities for troubleshooting purposes.

Cloud Services

a. Data Ingestion Service: Responsible for receiving data from the vehicle simulation in MATLAB Simulink. It should handle incoming data streams efficiently, perform initial validation, and pass validated data to the processing pipeline.

b. Data Validation Service: Validates the received data to ensure its integrity and authenticity. Any suspicious data should be flagged for further investigation.

c. Data Processing Service: Processes validated data for preparing it to enter the intrusion detection system. It can perform tasks like aggregation, filtering, and feature extraction to derive insights from the vehicle data.

Cloud Services

d. Security Service: Implements Intrusion Detection System to protect the cloud environment from false data injection attacks. Upon detecting a potential intrusion or false data injection attempt, the IDS microservice would trigger appropriate response mechanisms. This includes alerting system administrators and logging the event for further investigation.

Design Approach

Performing False Data Injection(FDI) Attack

The FDI Attack consists of the following steps:

(1) Determining the system boundaries and the target components.

(2) Defining severity metrics and fault models.

(3) Preparing a driving scenario-based test case.

(4) Performing tests by simulating FDI.

Design Approach

Performing False Data Injection Attack

(1) Determining the system boundaries and the target components.

The faults are injected via manipulation of the critical variables that represent the movement of the vehicles including:

- global longitudinal coordinate of vehicle i
- global lateral coordinate of vehicle i
- velocity of vehicle i
- acceleration of vehicle i
- yaw rate of vehicle i
- yaw angle of vehicle i
- relative distance between vehicle i and vehicle j
- relative velocity between vehicle i and vehicle j

Design Approach

Performing False Data Injection Attack

(2) **Defining severity metrics and fault models.**

Severity metrics:

- intended trajectory
- potential collisions with other vehicles or obstacle
- loss of control

Fault models:

- Manipulating sensor data: Injecting false sensor readings to deceive the vehicle's perception system.
- Manipulating control inputs: Altering the control commands sent to the vehicle's actuators, such as steering, braking, or acceleration.

Design Approach

Performing False Data Injection Attack

(3) Preparing a driving scenario-based test case.

Simulate a urban traffic scenario where the autonomous vehicle is surrounded by other vehicles. Then we inject false data to disrupt the vehicle's perception and control systems, potentially causing unsafe driving behavior.

(4) Performing tests by simulating FDI.

The faults are injected by aggravating their parameters in each simulation for the same driving scenario until collisions happen.

Finally, we can compare the FDI Attack Scenario and non-attack scenario.

Design Approach

Detecting the Attack

If an intrusion is detected that indicates potential false data injection, it's crucial to take immediate action to ensure the safety and integrity of the autonomous vehicle network.

Alerting Mechanism:

Automated Alerts: Set up automated alerts triggered by ML model outputs, notifying designated driver upon detection of suspicious sensor data.

Design Approach

Detecting False Data Injection Attack:

Detection Strategy:

- Data Analysis: Collecting sensor data transmitted from multiple autonomous vehicles to the cloud.
- Similarity Checks: Performing statistical analysis to identify patterns and deviations in the collected data.
- Feature Engineering: Use ML techniques to extract relevant features from the sensor data, capturing underlying patterns and relationships.

Design Approach

Detecting False Data Injection Attack:

- Intrusion Detection with ML: Training ML models (e.g., Isolation Forest, One-Class SVM, Autoencoders) on historical sensor data to learn normal behavior.
- Deviation Threshold: Setting up a threshold for acceptable data variations.
- Intrusion Detection: We will flag data points that significantly deviate from expected norms as potential false data injections.

Design Approach

Benefits of this approach:

- Improved System Reliability
- Reduced Safety Risks
- Building Customer Trust

Drawbacks of this approach:

- It can be complex and expensive to implement.
- It may not be foolproof, and there is always a risk that a sophisticated attacker could bypass the security measures.

Design Constraints & Dependencies

Constraints:

- Computational Resources
- Latency Requirements
- Privacy Concerns

Assumptions:

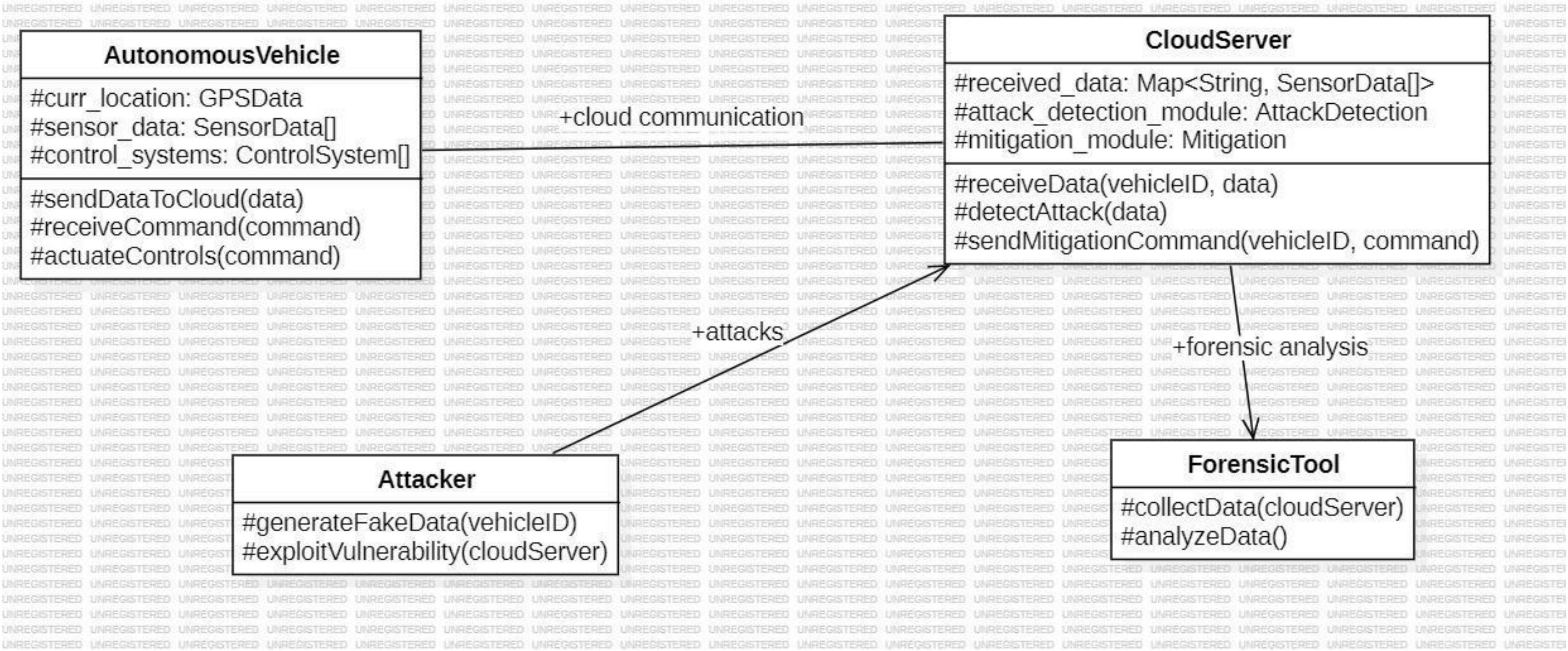
- Tamper-proof On-Board-Unit (OBU)
- Secure cloud environment
- Attack vector in V2C env. only

Design Approach Dependencies:

The success of the proposed design approaches relies on several dependencies with potential impacts:

- Successful set-up of the environment
- Successful attack execution
- Successful mitigation of the attack

Design Description



AutonomousVehicle
#curr_location: GPSData
#sensor_data: SensorData[]
#control_systems: ControlSystem[]
#sendDataToCloud(data)
#receiveCommand(command)
#actuateControls(command)
+cloud communication
CloudServer
#received_data: Map<String, SensorData[]>
#attack_detection_module: AttackDetection
#mitigation_module: Mitigation
#receiveData(vehicleID, data)
#detectAttack(data)
#sendMitigationCommand(vehicleID, command)
+attacks
+forensic analysis
Attacker
#generateFakeData(vehicleID)
#exploitVulnerability(cloudServer)
ForensicTool
#collectData(cloudServer)
#analyzeData()
UNREGISTERED

Project Progress

What is the project progress so far?

- Literature survey.
- Setting up vehicle environment.
- Running MOBATSim simulator.
- Enabling OpenStreetMap in MOBATSim
- Checking cloud connectivity of MOBATSim
- Collecting time-series dataset from vehicles that can be sent to cloud

Technologies used

- MOBATSIM Simulator
- SIMULINK
- MATLAB
- AWS/MICROSOFT AZURE
- JUPYTER / GOOGLE COLAB FOR TRAINING THE MACHINE LEARNING MODELS

References

Zhao, C., Comert, G., & Pisu, P. (2021). Secure Connected and Automated Vehicles against False Data Injection Attack using Cloud-based Data Fusion.

Zhao, C., Gill, J. S., Pisu, P., & Comert, G. (2021). Detection of False Data Injection Attack in Connected and Automated Vehicles via Cloud-Based Sandboxing.

Rosenstatter, T., Olovsson, T., & Almgren, M. (2021). V2C: A Trust-Based Vehicle to Cloud Anomaly Detection Framework for Automotive Systems.

Chowdhury, M., Islam, M., & Khan, Z. (2020). Security of Connected and Automated Vehicles.

Gao, K., Cheng, X., Huang, H., Li, X., Yuan, T., & Du, R. (2022). False Data Injection Attack Detection in a Platoon of CACC in RSU

Koley, I., Adhikary, S., Rohit, R., & Dey, S. (Year). A Framework for Evaluating Connected Vehicle Security against False Data Injection Attacks.

Saraoglu, Mustafa, et al. “MOBATSim: MOdel-Based Autonomous Traffic Simulation Framework for Fault-Error-Failure Chain Analysis.” IFAC-PapersOnLine, vol. 52, no. 8, Elsevier BV, 2019, pp. 239–44, doi:10.1016/j.ifacol.2019.08.077.

Thank You