Virtualization for Automotive Safety and Security Exploration

Md Rafiul Kabir and Sandip Ray
Department of ECE, University of Florida, Gainesville, FL 32611, USA.
kabirm@ufl.edu, sandip@ece.ufl.edu

2023 IEEE 16th Dallas Circuits and Systems Conference (DCAS) | 979-8-3503-9918-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/DCAS57389.2023.10130221

***Abstract*—A modern automobile system is a safety-critical distributed embedded system that incorporates more than a hundred Electronic Control Units, a wide range of sensors, and actuators, all connected with several in-vehicle networks. Obviously, integration of these heterogeneous components can lead to subtle errors that can be possibly exploited by malicious entities in the field, resulting in catastrophic consequences. We develop a prototyping platform to enable the functional safety and security exploration of automotive systems. The platform realizes a unique, extensible virtualization environment for the exploration of vehicular systems. The platform includes a CAN simulator that mimics the vehicular CAN bus to interact with various ECUs, together with sensory and actuation capabilities. We show how to explore these capabilities in the safety and security exploration through the analysis of a representative vehicular use case interaction.**

I. Introduction

We are experiencing an era of digital transformation at a societal scale, with the advancement of the Internet-of-Things, revolutionizing the notion of traditional embedded systems with pervasive connectivity and artificial intelligence [1]. One upshot of this transformation is our increased vulnerability to electronic and software failures and cyberattacks. It is obviously crucial for our well-being to develop technologies for the systematic exploration of the safety and security of critical cyber-physical systems.

Automotive systems constitute the quintessential representatives of critical cyber-physical systems with strong safety and security requirements. A modern automobile has hundreds of electronic control units (ECUs), several sensors and actuators, hundreds of megabytes of software, and several in-vehicle networks. Automotive systems also include interfaces to communicate with the external world. These components leave open a variety of opportunities for compromising the safety and security of the vehicle, possibly resulting in catastrophic accidents. A key issue is the introduction of connectivity into a congregation of components that were not originally designed with connectivity in mind. For instance, ECUs were originally designed to receive commands from the systems with which they communicate and to share information with any other hardware on the same CAN bus without the need for authentication and validation. However, with connectivity, they are open to exploitation by hackers. Correspondingly, sensors and actuators that contribute to the peripheral activities for ECU performance and are also prone to failure or adversarial compromise.

*This research has been supported in part by the National Science Foundation under Grant No. CNS-1908549.

In this paper, we demonstrate ways to explore the safety and security of automotive systems. We develop a prototyping infrastructure for exploring component failures and security compromises. We demonstrate the framework using a representative use case: the (right) turning functionality of the electronic power steering.

Our approach builds upon previous work on developing virtual prototyping infrastructure for automotive systems [2]. The infrastructure, ViVE, enables the exploration and coordination of different vehicular components while abstracting detailed functionality of ECUs, sensors, actuators, etc. However, security and safety were not considered, the focus was on exploration and optimization. In this paper, we extend that infrastructure to account for component failures and adversarial attacks that may result in critical use case dysfunction.

II. Related Work

Virtualization of automotive components has become a viable topic to explore the possibilities of development in vehicular electronics. In order to manage multiple in-vehicle execution software, Lee *et al.* [3] introduced a Virtualized Automotive Display (VADI) system for a digital cluster. Safar *et al.* [4] proposed the integration of a virtual platform (VP) with the V-model of automotive software development to facilitate verification and validation at the SoC, ECU, and system level. Strobl *et al.* [5] presents the benefits of automotive virtualization as a foundation for consolidating a large number of ECUs into a small number of Domain Controller Units (DCUs). There are a few automotive simulators and security exploration platforms for domain-specific usage. SUMO [6] acts as an open-source miniature traffic simulation platform. CARLA [7] enables the development, training, and validation of autonomous driving systems. In order to conduct multi-vehicle experiments when real vehicles are not readily available, Yang *et al.* [8] designed a digital twin prototype. For a hands-on security exploration, AutoHaL [9] emerged as an effective platform that explored ranging sensor attacks. Scalas *et al.* [10] suggested a systematization of knowledge regarding fundamental cybersecurity factors to consider when developing a modern automobile.

979-8-3503-9918-9/23/$31.00 ©2023 IEEE

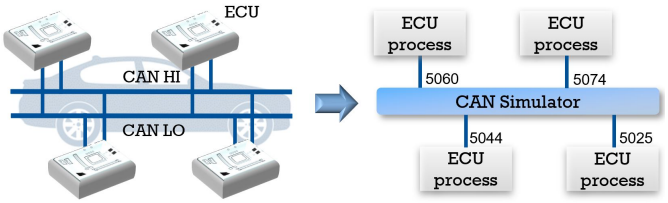

Figure 1. Platform provided CAN simulator modeling

III. System Design and Use Case Implementation

The ViVE platform enables the exploration, optimization, and exercise of various security targets, with system-level coordination involved in various use cases. While the specifics of vehicular components (beyond the capability required to comprehend the use case) can be abstracted, vehicular communications, in particular, are incorporated. Here we briefly discuss the component models and the CAN simulator components, which are crucial to the extension of ViVE to incorporate functional safety and security. We also introduce the "right turn" use case.

A. ViVE *component models*

Automotive systems typically consist of sensors, actuators, and ECUs connected to an in-vehicle communication network (e.g., direct electrical signal, CAN, etc.). We virtualize all these components in our simulated environment to exercise use cases and security scenarios.

- *ECU:* ViVE does not require a complete software model of the ECU, in contrast to conventional simulation platforms. Instead, it utilizes the computational capabilities of an actual ECU to simulate functionality that is pertinent to the use cases by receiving inputs from appropriate sensors or other simulation blocks.
- *Sensors and Actuators:* Similar to ECUs, we offer an interface that may be used to connect a physical sensor or actuator, as well as a software process that creates synthetic sensory or actuarial data.

The goal of ViVE is to provide the user with a realistic understanding of the interaction of various components and subsystems via automotive use cases. As a result, all of the vehicular components are represented as continuously running computation blocks. For instance, the brake pedal position sensor continuously sends brake input data to the ABS ECU (part of the ABS use case).

B. *CAN Simulator*

In order to enable the coordination and communication of automotive components we provide an in-vehicle communication system that allows two types of communication:

1) Electrical signal (without ECU involvement or CAN)
2) CAN communication (among ECUs)

We incorporate a versatile, reconfigurable communication simulator mimicking the activities of an actual automotive CAN bus, referring to it *CAN simulator.* The simulator implements interaction between all of the ECU activities using Transmission Control Protocol (TCP) sockets, with socket programming based on a common client-server model. The CAN frame is created as a byte array message for data transmission and receipt over the socket and is modeled with a Python-CAN package. The message frame includes the actual data as well as additional information, *e.g.*, the arbitration ID, extended ID, and data length. The CAN simulator can alert the user whenever there is a disruption in the network. Fig. 1 shows the virtualization from the CAN bus to the CAN simulator. Communication with ECU processes is established with corresponding TCP socket port numbers.

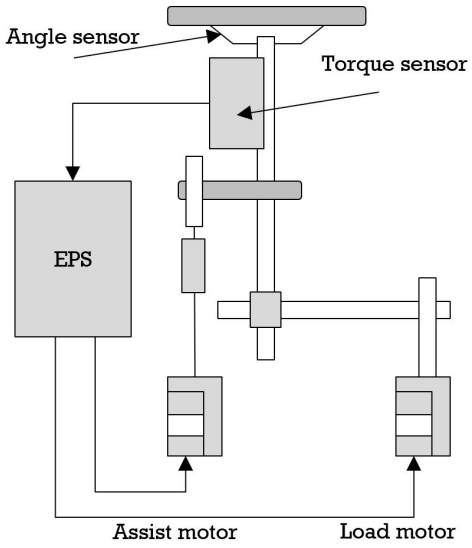

Figure 2. Primary component setup used as a basis to implement the Right Turn use case

C. *Right Turn Use Case*

The electric power steering (EPS) system facilitates the turning of steering wheels for the basic operational control of a car. *Right Turn* is a segment of the operation that occurs when the driver turns his steering wheel to the right. The EPS system helps drivers steer the vehicle with minimal effort. The functionality is fairly standard [11]. The use case simulated components are indigenous processes for the ECUs, sensors, and actuators, *i.e.*, EPS ECU, ABS ECU, gateway ECU, angle sensor, torque sensor, assist motor, and load motor. The user initiates the Right Turn use case by turning the steering wheel to the right *i.e.*, sending steering wheel sensory data, which is enabled via the ViVE GUI. All the communications are implemented as byte array messages (via socket) by each component (see Table I). Each computation process is associated with arbitration IDs (*e.g.*, 1, and 3) and port numbers (*e.g.*, 5004, 7400, etc.). The primary functionality of this use and the corresponding system in our platform are shown, respectively, in Fig. 3 (a) and (b).

On the action, the system behaves as follows:

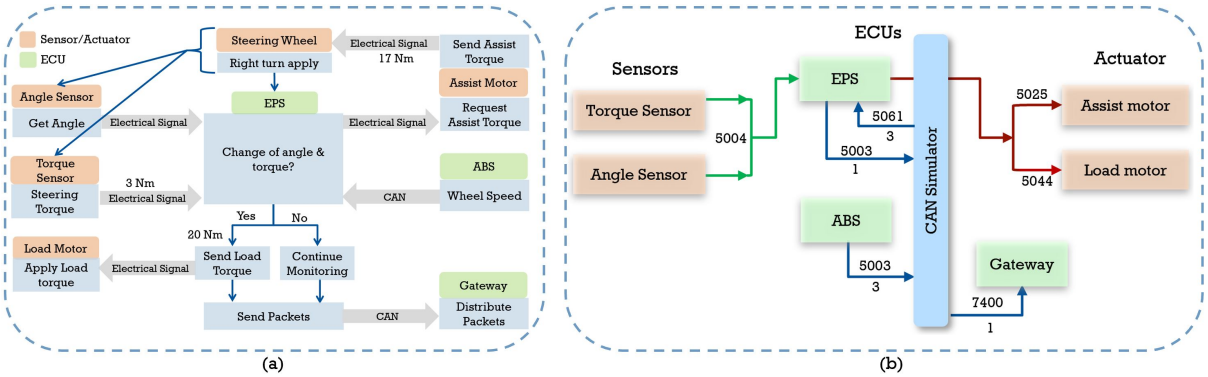

Figure 3. Right Turn (from EPS) use case (a) functionality flow diagram and (b) platform's primary system design

Table I
RIGHT TURN USE CASE BYTE ARRAY MESSAGES

Components	Data
Angle sensor	[0, 90, 180, 270, 360]
Torque sensor	[0] or [3]
ABS	simulated values from [0] to [70] (CAN)
EPS (output-1)	calculated value between [0] and [17]
EPS (output-2)	calculated value between [0] and [20]
EPS (for gateway)	output-1 and output-2 values (CAN)

- In order to simulate the driver making the right turn, the torque sensor and angle sensor data are continuously sent to the EPS ECU. For simulation purposes, we assume the steering torque to be 3 Nm and various angles of the steering wheel as 0, 90, 180, 270, and 360 degrees.
- Additionally, the EPS ECU receives wheel speed data as a CAN message from the ABS ECU, which is required for the assist torque calculation.
- Depending on all three sensory data, the EPS ECU calculates the assist torque required. For instance, if the load torque required is 20 Nm (varies for different speed values) then the assist torque would be (20 - 3) = 17 Nm
- Then, the EPS ECU sends assist torque and load torque to the actuators: assist motor and load motor, respectively.
- Furthermore, the EPS ECU sends a CAN frame back to the CAN simulator with an arbitration id 1, denoting the gateway ECU.

IV. SECURITY COMPROMISE EXPLORATION

The initial phase of any adversarial attack focuses on infiltrating the in-vehicle network, by typically accessing the onboard diagnostics (OBD) or wireless interfaces [12]. Nevertheless, there are several other works that explored different entry points *e.g.*, through a modified WMA audio file [13], a USB-connected smartphone [14], etc. Irrespective of the attacker entry point, the goal is to get access to the CAN bus and consequently infiltrate the CAN frame, which will disrupt major automotive ECU functions. Here we extend VIVE to enable the exploration of two modes of attacker entry. These compromises enable a security architect to explore the possible extent of system compromise in the context of a cyberattack.

Frame Falsification: If the specifics of the CAN frames are known, the adversary can falsify the CAN messages with fake data to deceive the ECUs *e.g.*, false data from the wheel speed sensor in the case of the Right Turn use case. In section III we mentioned the importance of the wheel speed data in order to get correct assist torque values for the steering wheel. Let us assume, the adversary has falsified the CAN messages coming from the ABS to the EPS ECU. In this situation, the EPS calculation will provide incorrect torque values for the steering wheel as well as the car wheels. In the real-life scenario, the driver can face multiple critical safety-critical problems: 1) the driver may not steer the steering wheel properly, resulting in the car not making a right turn at all, 2) the car may make the right turn so fast that it can lose control and fall into a serious accident. Our platform enables exploration of this scenario via real-time simulation (Fig. 4).

DoS Attack: We extend the CAN simulator to include arbitration ids and priority schemes to employ queuing mechanisms, *i.e.*, prioritizing which ECU messages to transmit first. When a high-priority frame is transmitted, lower-priority ones are forced to delay, allowing Denial of Service (DoS) attacks to occur. Let us assume, the adversary has implemented a DoS attack by flooding the CAN simulator with high-priority messages coming solely from the EPS ECU. In that case, the CAN simulator will not receive anything from the ABS ECU, resulting in a lack of updated wheel speed data. In this regard, similar safety-critical scenarios will occur discussed for frame falsifying.

V. FUNCTIONAL SAFETY EXPLORATION

Automotive functional safety ISO 26262 standards were prepared to address the risks as failures occur. There are electronic component failure rates, failure classifications, and hardware failure modes to understand the impact and severity of the failures. With that being said, there are still no exploration platforms for exploring system-level failures in a virtual prototyping environment. We allow seamless component swapping and explore use cases to simulate component failures. Moreover, VIVE allows hardware integration with real sensors and ECU process implementation in small-scale computers

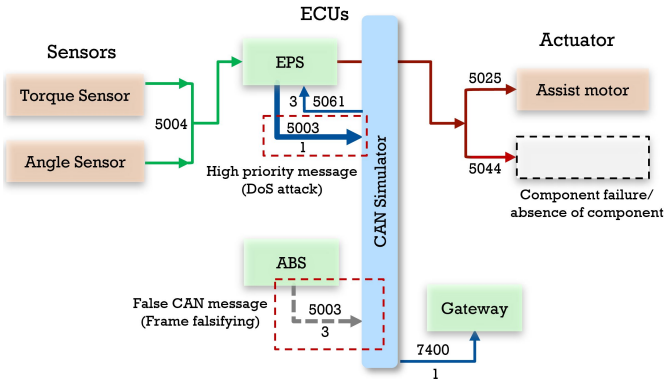

Figure 4. Safety and security exploration in Right Turn use case with VIVE

(*e.g.*, raspberry pi) to explore real-time failure explorations. But, within the virtual environment, this exploration is less sophisticated and more easily implementable to exercise and understand the safety critical situation due to failures.

We extend VIVE with the ability to "tweak" any components (*i.e.*, ECU, sensor, or actuator) in a seamless configurable manner from any use case and explore the corresponding result of that use case performance under critical component failure with a variety of failure modes. Fig. 4 depicts the scenario where the load motor actuator has either (1) completely failed, or (2) lost electrical connection with the EPS ECU. Therefore, we are simulating the use case with port number 5044 being connected to a blank process. As a result, the final output will generate an error saying the wheels are not turning with adequate movement because no torque is being applied. In a real-life scenario, the driver will not be able to make a successful right turn with completely zero or partial (considering a minimal amount of steering torque coming from the driver) wheel movement. Similarly, an ECU malfunction will result in a major safety-critical problem, *i.e.*, severe dysfunction in ECU computations.

VI. Countermeasures

We explored a few security scenarios within our virtual environment to show its viability and exploration capabilities. Obviously, VIVE only provides an infrastructure for exploration, and understanding of failure, not a resiliency solution. Nevertheless, the user can define countermeasures and explore their efficacy with VIVE. Major countermeasures against CAN compromises include dedicated hardware, cryptography, intrusion detection, access control, etc. [10]. Within our platform architecture, the CAN simulator and ECU processes are capable of alerting users at the initial stage of certain component failures and abnormal CAN message structures. We employ the usage of arbitration IDs and TCP infrastructure and infused basic detection capabilities.

VII. Conclusion

Exploring and exercising system-level functional safety and security scenarios early in the design is a critical requirement for system-level validation of automotive systems. In this paper, we addressed this problem through a platform to explore it based on real-life attack occurrences and failure modes. We discussed how to implement these features on top of an existing prototyping platform VIVE that was originally designed for functionality exploration and optimization. We used a representative use case, *right turn*, to demonstrate the viability of the approach.

In future work, we aim to implement more safety and security scenarios and the corresponding countermeasures. For instance, creating use cases to establish the functional safety concept and consequently explore the viability of hardware and software safety requirements is a critical exploration area. Furthermore, adversarial cyber attacks on computer vision modules, machine learning mechanisms, and vehicular communications are potential exploration challenges to be considered for VIVE.

References

[1] L. Da Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on industrial informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[2] M. R. Kabir, N. Mishra, and S. Ray, "Vive: Virtualization of vehicular electronics for system-level exploration," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2021, pp. 3307–3312.

[3] C. Lee, S.-W. Kim, and C. Yoo, "Vadi: Gpu virtualization for an automotive platform," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 1, pp. 277–290, 2015.

[4] M. Safar, M. A. El-Moursy, M. Abdelsalam, A. Bakr, K. Khalil, and A. Salem, "Virtual verification and validation of automotive system," *Journal of Circuits, Systems and Computers*, vol. 28, no. 04, p. 1950071, 2019.

[5] M. Strobl, M. Kucera, A. Foeldi, T. Waas, N. Balbierer, and C. Hilbert, "Towards automotive virtualization," in *2013 International Conference on Applied Electronics*. IEEE, 2013, pp. 1–6.

[6] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "Sumo–simulation of urban mobility: an overview," in *Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation*. ThinkMind, 2011.

[7] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[8] C. Yang, J. Dong, Q. Xu, M. Cai, H. Qin, J. Wang, and K. Li, "Multi-vehicle experiment platform: A digital twin realization method," in *2022 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2022, pp. 705–711.

[9] B. B. Y. Ravi, M. R. Kabir, N. Mishra, S. Boddupalli, and S. Ray, "Autohal: An exploration platform for ranging sensor attacks on automotive systems," in *2022 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE, 2022, pp. 1–2.

[10] M. Scalas and G. Giacinto, "Automotive cybersecurity: Foundations for next-generation vehicles," in *2019 2nd International Conference on new Trends in Computing Sciences (ICTCS)*. IEEE, 2019, pp. 1–6.

[11] J.-H. Kim and J.-B. Song, "Control logic for an electric power steering system using assist motor," *Mechatronics*, vol. 12, no. 3, pp. 447–459, 2002.

[12] J. Liu, S. Zhang, W. Sun, and Y. Shi, "In-vehicle network attacks and countermeasures: Challenges and future directions," *IEEE Network*, vol. 31, no. 5, pp. 50–58, 2017.

[13] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno *et al.*, "Comprehensive experimental analyses of automotive attack surfaces." in *USENIX security symposium*, vol. 4, no. 447-462. San Francisco, 2011, p. 2021.

[14] S. Mazloom, M. Rezaeirad, A. Hunter, and D. McCoy, "A security analysis of an in-vehicle infotainment and app platform." in *WOOT*, 2016.