Analyzing the Impact of Injected Sensor Data on an Advanced Driver Assistance System using the OP$_2$TIMUS Prototyping Platform

Alexander Stühring*, Günter Ehmen*, Sibylle Fröschle†

*University of Oldenburg, Germany
{stuehring|ehmen}@informatik.uni-oldenburg.de

†OFFIS, Germany
sibylle.froeschle@offis.de

***Abstract*—Modern vehicles are running complex and safety critical applications distributed over several Electronic Control Units (ECUs). Some ECUs are equipped with communication interfaces providing access to other devices, networks or remote services. Since the number of attack vectors is increasing, an early investigation of the impact of attacks becomes steadily more important. This paper gives an example how manipulated sensor data injected to the CAN bus affects an Advanced Driver Assistance System (ADAS). Within multiple experiments we illustrate the impact of different aspects like the sending rate.**

***Index Terms*—Impact Analysis, Automotive, Security, Safety-Critical, Cyber-Physical Systems, Hardware-In-the-Loop**

I. Introduction

Modern vehicles have approximately 50 to 70 ECUs such as the engine-control operating the fuel injection and the ignition system. In addition, ECUs communicate with each other via in-vehicle networks like LIN, CAN or FlexRay to realize distributed functions. Therefore, a strict separation of safety-critical and non safety-critical networks is not possible. This creates new opportunities for attackers. It was already shown in 2010 that an intruder could send messages to a safety-critical CAN bus by exploiting vulnerabilities in the software of an ECU [1]. At first sight it appears that getting access to in-vehicle networks requires physical access. However, some ECUs provide a link from in-vehicle networks to wireless interfaces like Bluetooth. If such a component is compromised, it is possible to manipulate the vehicle without having physical access to it [2]. Access to the in-vehicle networks grants an attacker various options to manipulate the communication between the ECUs [3]. Previous works often focused on manipulating the behavior of real automotive hardware in a laboratory environment [4]. However, they did not consider the impact on the vehicle or on the driving behavior. An exception is provided by Koscher *et al.* which investigated the impact on a car while driving on an airport runway under rigorous safety requirements [1].

In this paper we analyze the impact of manipulated sensor data on an ADAS. This is accomplished by injecting and tracing manipulated data from network to task level while monitoring driving and system behavior. In contrast to state-of-the-art, OP$_2$TIMUS (**O**pen **P**rototyping **P**latform for **T**esting, **S**imulating and **M**easuring Distrib**u**ted Embedded **S**ystems) allows to utilize the benefits of Hardware-in-the-Loop (HIL) simulations coupled with traffic simulations and thus enables the investigation of attacks on ECUs in different scenarios without having to perform physical tests with real cars.

Section II introduces the prototyping platform, the ADAS and the traffic simulator. Section III investigates the impact of manipulated sensor data, explains the different experiments and results. Finally, Section IV sums up the results.

This work was supported by the funding initiative *Niedersächsisches Vorab* of the Volkswagen Foundation and the Ministry of Science and Culture of Lower Saxony as part of the *Interdisciplinary Research Center on Critical Systems Engineering for Socio-Technical Systems*.

II. Introducing the Prototyping Platform

OP$_2$TIMUS allows to analyze complex systems in the automotive domain and is equipped with different ECU boards and communication buses to provide a variable configuration of several system architectures (see Fig. 1). Additional analysis boards enable runtime measurement and recording of network traffic. Complex automotive scenarios can be analyzed in a residual bus simulation or in a HIL simulation in conjunction with a simulation environment which is connected to the CAN bus.

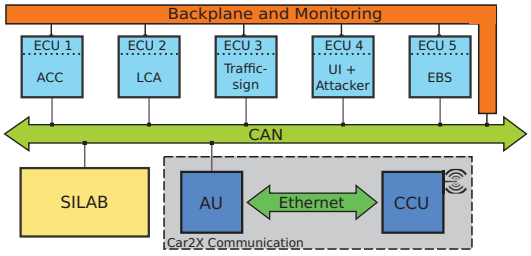

Fig. 1: Experimental setup with simulation environment.

A. Hardware and Software Components

1) ECUs: The prototyping platform comprises five ECU boards, each equipped with an ARM7 processor, a Field Programmable Gate Array (FPGA), two CAN, and two FlexRay bus connectors. A configurable operating system based on the OSEK standard allows a flexible application of the ECUs. The FPGAs provide LIN functionalities through a dedicated IP core and allow the usage of customized network controller designs. Moreover, in order to investigate in-vehicle security aspects the operating system was extended [5]. This update

allows the manipulation of the communication between tasks as well as between tasks and bus-systems.

2) Backplane and Monitoring: All components of the prototyping platform are connected via a backplane which provides a centralized configuration and programming interface to the ECUs and the monitoring boards. The monitoring boards allow recording of bus traffic and code instrumentations for a measurement based execution time analysis. In combination with the OSEK-based operating system and its extensions the monitoring is able to track execution behavior of tasks as well as messages through the protocol stack. This allows to determine the exact time a task computes data or receives interrupts from the hardware.

3) Car2X Communication: The Car2X extension integrates hardware and software components to enable vehicular communication based on IEEE 802.11p. The extension comprises an Application Unit (AU) and a Communication and Control Unit (CCU). The AU implements the Car2X applications and creates a connection between the wireless interfaces of the CCU and the in-vehicle networks like CAN or FlexRay.

B. The ViDAs Benchmark

Implemented as a MATLAB / Simulink / Stateflow model, the Virtual Driver Assistance (ViDAs) benchmark comprises an ADAS containing Adaptive Cruise Control (ACC), Lane Change Assistant (LCA) and traffic sign recognition functionalities. By means of integrated vehicle sensors, the system is able to detect and visualize traffic signs of speed limitations and restrictions on overtaking which are installed at the roadside. At any time, the driver is capable of overriding or deactivating the ADAS.

ACC: The ACC continuously maintains a speed dependent safety gap to a preceding vehicle. Simultaneously, a constant velocity selected by the driver is maintained as long as a preceding vehicle is not driving at a lower velocity than the ego-vehicle. An identified speed limitation may also restrict the desired velocity. Activation of the ACC as well as selecting the desired velocity is provided by dedicated operation elements. A light signal represents the activation of the velocity option while the selected velocity is being indicated on a display.

LCA: By means of light signals installed in the exterior mirrors, the driver is informed whether a lane change is currently possible without violating the safety gap to other traffic participants. In doing so, only the driving lane next to the ego-vehicle is regarded. A green light signal symbolizes the opportunity of a safe lane change. Otherwise a red signal indicates that a lane change is currently not possible due to risks or restrictions affecting the traffic safety.

C. Simulation Environment

The driving simulation software SILAB[1] allows for the ViDAs benchmark execution within a simulation environment. The ego-vehicle is represented by a middle-class vehicle - a Skoda Octavia - as it provides defined dimensions and is available as a model for the simulator. In addition, the required sensors have been implemented within the simulator. These sensors are idealized so that noisy sensor data is non-existent.

[1]Würzburg Institute for Traffic Sciences, Website: https://wivw.de/en/silab/

III. Impact of Manipulated Sensor Data

An ADAS relies on correct and non-manipulated sensor data in order to work correctly. But what happens if an intruder manages to insert manipulated sensor data into the system? Assume there is a bus system with connected ECUs and one of these ECUs is compromised. It is clear that this can lead to various attacks such as that the attacker can flood the bus in order to block the bus. It is, however, less clear how an attacker can influence functionality such as an ADAS in a specific way. This is because the impact of injected messages will depend on the interplay between the injected messages and the messages of the honest components.

A. Experimental Setup

The experimental setup comprises three different components: the benchmark, the simulation environment and the driver. The ViDAs benchmark is split into five dedicated tasks and mapped on the available ECUs of OP$_2$TIMUS (see Fig. 1). The simulation environment provides the ego-vehicle with its driving physics as well as an environment including other vehicles and traffic participants. Moreover, SILAB offers an interface for the driver as well as a CAN bus connector which allows the integration of OP$_2$TIMUS. In case of an active manipulation, the driver commonly recognizes that something is out of the ordinary, thus disabling the ADAS manually. The scenario focuses on manipulated sensor and controller values and leaves the driver behavior to future work. Fig. 2 illustrates the communication flow between the different components. First, the ego-vehicle is controlled by the driver via control elements. Next, the simulation environment simulates the different components and calculates the corresponding driving behavior. The ADAS processes the sensor data and sends the control signals back to the simulation environment where the driving behavior is calculated. Finally, informations like the current speed are displayed via a dash-board to the driver.

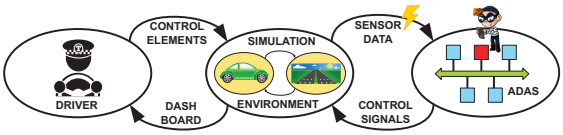

Fig. 2: Communication flow between driver, simulation environment and ADAS / OP$_2$TIMUS.

B. Scenario

The scenario assumes that an attacker was able to compromise ECU 4 which provides the user-interface and gained access to the CAN bus in order to manipulate the ACC. The simulation itself is configured to simulate the ego-vehicle on an endless road. The driver accelerates the car up to 40 km/h and activates the ACC speeding up the car to a pre-configured speed of 120 km/h. From that moment OP$_2$TIMUS tracks the speed of the ego-vehicle as well as the brake and acceleration

signals from the ACC. The injected sensor data mimic a parking vehicle directly in front of the ego-vehicle to achieve the greatest possible impact on the controller and therefore on the speed which defines the impact within this scenario. In order to compare the results, the setup was configured to run one simulation with a real parking vehicle direct in front of the ego-vehicle. This data is used in the following as comparative simulation and defines the maximum impact on the system.

C. Experiments

1) Determinism of Attacks: Within this first experiment, the attacker is configured to start injecting manipulated sensor data for five seconds as soon as the ego-vehicle reaches the target speed of 120 km/h. The simulation environment as well as the attacker send their messages with a fixed period of 30 ms. The simulation is repeated 20 times and the results show the affecting speed of the ego-vehicle in Fig. 3, illustrated as dashed lines. In addition, the comparative simulation results are shown as solid line. The impact of the

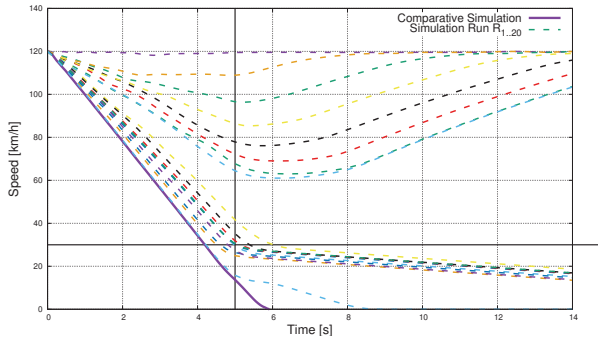

Fig. 3: Not deterministic impact on the speed while injecting manipulated sensor data to the ACC.

compromised sensor data is not deterministic. The extreme points in the behavior observed in the simulation range from instantaneously braking to towards no reaction at all. Since the software of all ECUs is reseted synchronously before each simulation run, the reason for this not deterministic behavior must reside in the communication behavior.

2) Communication Behavior: The second experiment is similar to the first one. But this time the focus is on the communication and task activation behavior of the ACC. Fig. 4 shows an excerpt of the recorded timing information of the two simulation runs from the previous experiment. The figure illustrates task activation and message parsing behavior of the simulation run with the lowest impact R_1 and with the highest impact R_2 on the driving behavior. In the following, received messages with non-manipulated sensor values are labeled m' and received messages with manipulated sensor values m''. T^{ISR} describes the time sequence of incoming CAN interrupts which are traced with code annotation within the interrupt service routine (ISR). Analogous to this, T^{PS} describes the message task activation which is located in the protocol stack and T^{ACC} describes the activation of the ACC task. As depicted in both runs, the attacker's message m'' is received by the

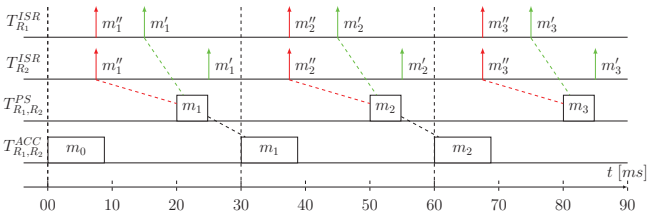

Fig. 4: Timings of received CAN messages, message task and ACC activation.

ACC before the original message m'. In contrast to R_2, the original message was received within R_1 before the protocol stack was executed. In this case, the protocol stack will reject the older message m'' with the manipulated sensor data to ensure that always the newest data is provided to the tasks. If the timing is constant the injected messages have no effect. Conversely, this also means that the original message m' will be rejected and will not effect the task if it is received too late. The reason for the different results can be found in the timing behavior of the original sensor data. Since SILAB does not restart at exactly the same time, the communication behavior differs in every simulation run. If the attacker has no knowledge about the timing of the sensor data on the bus he has no chance to cause a deterministic impact on the ACC controller. However, if an attacker has knowledge about the timing of the messages on the CAN bus, he can try to place the message with manipulated sensor data directly after the original messages. This also explains why flooding is not an ideal approach for an attacker in this case. Since the message task is providing new sensor information periodically, there is no reason for an attacker to send manipulated sensor data with a lower period.

3) Sending Interval: The third experiment is designed to check whether the impact changes when an attacker is able to inject messages only with a higher period in contrast to the original sender. In comparison with the previous scenarios the underlying setup of the scenario was modified to create deterministic runs. This guarantees a nearly constant timing gap between original and manipulated sensor data. In the following, we distinguish between three different configurations. Configuration A (injecting messages every 30 ms) is equivalent to the attacker discussed in the previous section. Configuration B (injecting messages every 60 ms) and C (injecting messages every 90 ms) are sending manipulated sensor messages with a higher period time. Fig. 5 and Fig. 6 illustrate the correlation between the speed of the ego-vehicle, the braking signal, and the acceleration signal. Within the comparative simulation the ego-vehicle slows down very rapidly. After about 4.17 s the vehicle passes the 30 km/h barrier which deactivates the ACC. As illustrated in Fig. 5 the braking signal ranges over 100 %. The reason for this can be found in the comfort-mode limiting the negative acceleration value. In case of an emergency brake situation the ACC leaves the comfort mode which allows brake signals higher than 100 %. If the compromised ECU injects messages according to configuration A with a sending period of 30 ms, the vehicle needs 4.64 s to reach the 30 km/h

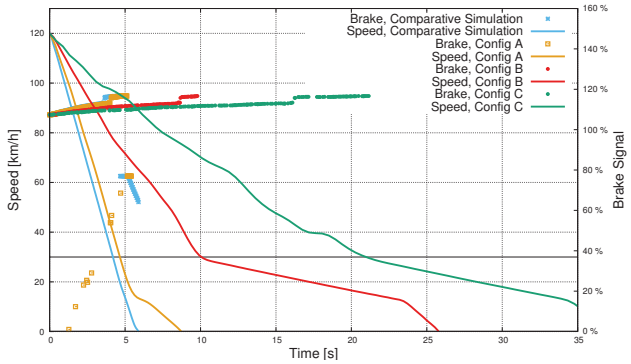

Fig. 5: Impact of compromised sensor data on speed and braking signal.

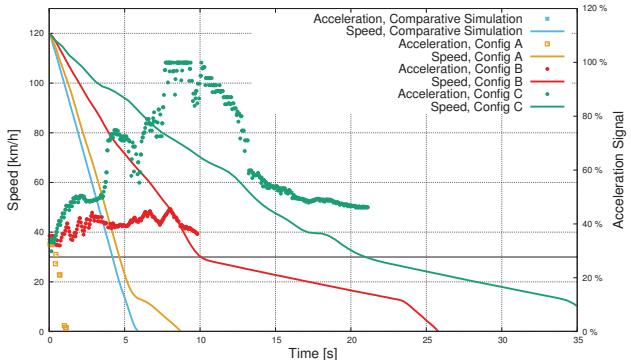

Fig. 6: Impact of compromised sensor data on speed and acceleration signal.

barrier. The reason for this difference in comparison to the comparative simulation is reasoned by fact that the original sensor data interferes with the compromised sensor data. The used approach for creating a deterministic simulation can not guarantee that all sensor data received by the ACC is manipulated. According to the results 93.48 % of all received and computed data is compromised. The original sensor data leads the ACC to take countermeasures, reduce the brake signal and increase acceleration. This effect is illustrated by single signal deviations in Fig. 5 and Fig. 6. In configuration B the attacker injects messages with a period of 60 ms and the vehicle decreases at a slower rate. The time for reaching the 30 km/h barrier doubled to 9.82 s compared to the comparative simulation. In contrast to the previous configuration it is important to note that the impact of injected messages on the ACC increases dramatically as illustrated by the acceleration signal increases dramatically in Fig. 6. According to the traces, the ACC received only original sensor data in 40.83 % of all cases. This effect increases by raising the injecting period to 90 ms in configuration C. The ego-vehicle reaches the 30 km/h barrier after about 20.92 s. Since the ACC receives only manipulated sensor in 27.1 % of all cases, the controller tries to counterbalance the different input stronger in comparison to the previous configurations.

D. Results

The investigation has shown that the impact of manipulated sensor data strongly depends on the exact time a message with manipulated sensor data is received by an ECU and processed by their tasks. Moreover, an attacker is only able to create deterministic impact on the driving behavior if he is able to take the communication behavior into account or if he injects manipulated sensor data in a significantly smaller period. Moreover, the experiments were able to estimate the impact of the sending period. TABLE I shows that increasing the transmission period doubles the time the car needs to reach the 30 km/h barrier.

TABLE I: Results of the different configurations.

Configuration	Impact (Time)	Received Brake Signals
A (30 ms)	4.64 s	93.48 %
B (60 ms)	9.82 s	40.83 %
C (90 ms)	20.92 s	27.10 %

IV. Conclusion

In this paper we introduced OP$_2$TiMuS and showed how the prototyping platform allows a fine-grained analysis of the impact of manipulated sensor data followed by three comprehensive experiments. We have shown that injected sensor data can have a huge impact on the driving behavior. The concrete impact strongly depends on the configuration of the protocol stack as well as the task activation behavior. Moreover, we have identified that an attacker has not necessarily to insert the manipulated messages faster than the original sender does. It is more important to place the message in the protocol stack after the original message but before task execution. Furthermore, we were able to observe an exponential correlation between the impact of manipulated sensor data and the sending period.

References

[1] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Experimental security analysis of a modern automobile," in *IEEE Symposium on Security and Privacy*, 2010.

[2] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive experimental analyses of automotive attack surfaces," in *Proceedings of the 20th USENIX conference on Security*, 2011.

[3] M. Wolf, A. Weimerskirch, and T. Wollinger, "State of the art: Embedding security in vehicles," *EURASIP Journal on Embedded Systems*, 2007.

[4] T. Hoppe, S. Kiltz, and J. Dittmann, "Security threats to automotive can networks — practical examples and selected short-term countermeasures," in *Proceedings of the 27th International Conference on Computer Safety, Reliability, and Security*, 2008.

[5] A. Stühring, "Untersuchung der angriffssicherheit von can-netzwerken im automobilbereich," Master's thesis, Carl von Ossietzky Universität Oldenburg, 2012.