www.dspace.com

WHITEPAPER

From HARA and TARA to Risk-Based Safety and Security Dependency Testing

Roman Trentinaglia, Dr. Markus Fockel
Fraunhofer Institute for Mechatronic Systems Design IEM

Dr. Matthias Pukrop, Tobias Schaeffer
dSPACE GmbH

Fraunhofer IEM

YOUR PARTNER IN SIMULATION AND VALIDATION

dSPACE

Modern vehicles are becoming more connected and autonomous, and more software-defined in general. Such connectivity leads to security risks due to the increased attack surface for external intrusions. In addition, attacks can also lead to safety hazards as cars contain multiple safety-critical components. Therefore both safety and security must be considered in combination. In this whitepaper, we describe a tool-supported analysis method aligned with automotive standards to identify safety and security dependencies and automatically derive corresponding test cases. These test cases can be imported into the existing dSPACE tool chain to improve efficiency by reducing time-consuming manual work and susceptibility to errors. Thereby, our method brings together system design and testing phases to pave the way for an integrated safety and security-by-design life cycle in the automotive domain.

Introduction

With the many already existing advanced driver assistance systems (ADAS) and upcoming autonomous functionalities, automated driving is becoming more prevalent. These functions are safety-critical, and failures can pose hazards and harm people. Security attacks on vehicles are also increasing every year [Upstream2024].
To ensure safety and security, measures must be implemented throughout the entire development process. Accordingly, ISO 26262 and ISO 21448 require safety measures to be taken, while ISO 21434 describes required security measures. These standards demand that companies establish processes to analyze safety and security hazards and threats, implement appropriate countermeasures, and verify and validate their effectiveness. If these processes are carried out in isolation from each other, there is a risk that dependencies and overlaps between safety and security are not considered.
Security attacks will lead to safety problems, so it is essential to consider both aspects together to achieve safety and security by design. But how can we implement processes and methods that leverage synergies and especially target the dependencies between safety and security? How can we find test cases that specifically address the conjunction of safety and security?
The development of one technical system requires one end-to-end process covering all relevant standards. In the presented approach, we use a safety and security dependency analysis to explicitly spot safety and security dependencies and automatically derive corresponding test cases in order to verify and validate the effectiveness of implemented countermeasures. In this way, the method helps to carry out a comprehensive process covering safety as well as security. In particular, we describe a tool-supported method to identify safety and security dependencies, derive risk-based test cases, and automate according XIL test configuration and execution. Our tool-support method also enables the automated generation of trace links (which we will not go into further detail in this whitepaper). Figure 1 shows an overview of the method. Hazards and threats identified during HARA and TARA are fed into Fraunhofer IEM's automated safety and security dependency analysis [Foc2022]. This analysis automatically derives combinations of faults and security attacks that can lead to hazards, taking into account existing safety and security mechanisms designed into the system of interest. The safety and security engineering goal is to make these so-called minimal cut sets (MCS) as big as possible, indicating that only the combination of many events (attacks and failures) at the same time can cause a hazard to occur. Based on the MCS, i.e., the critical attacks and faults, we automatically derive a list of test cases. These test cases check if the events contained in an MCS truly must occur in combination or if a smaller set of events can already lead to a hazard. The latter would indicate that a safety or security mechanism is not working as intended. As a final step in our method, the test case list is used to automatically configure and execute XIL tests for the system of interest.

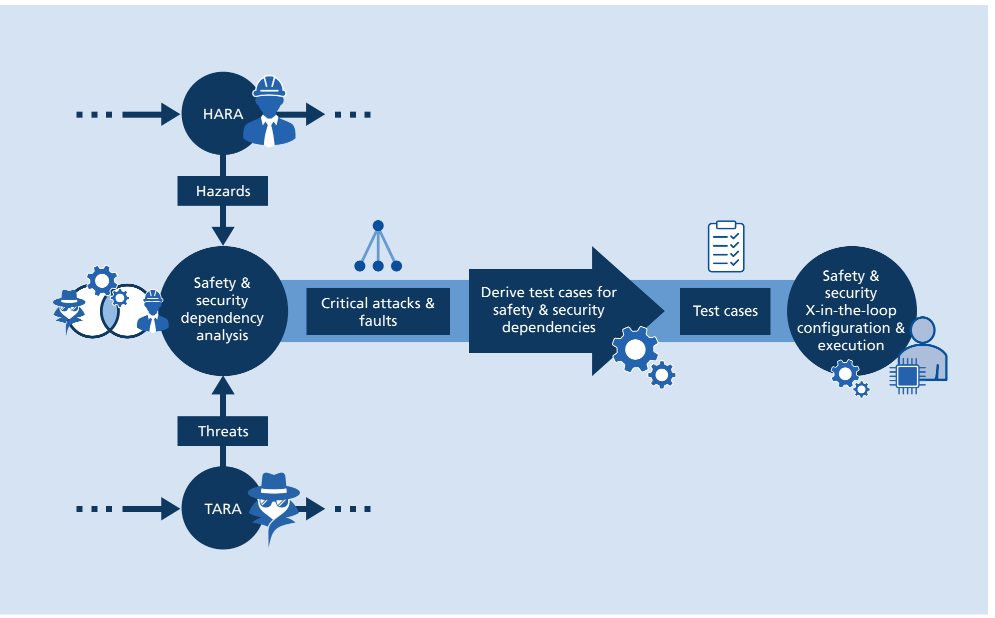

HARA
Hazards
Safety & security dependency analysis
Critical attacks & faults
Derive test cases for safety & security dependencies
Test cases
Safety & security X-in-the-loop configuration & execution
Threats
TARA

Figure 1: Schematic view of the steps for our approach. Existing HARA and TARA results are inputs of our safety & security dependency analysis which is able to automatically derive test cases for relations between hazards and threats. These test cases can be automatically executed in the existing dSPACE X-in-the-loop tool chain.

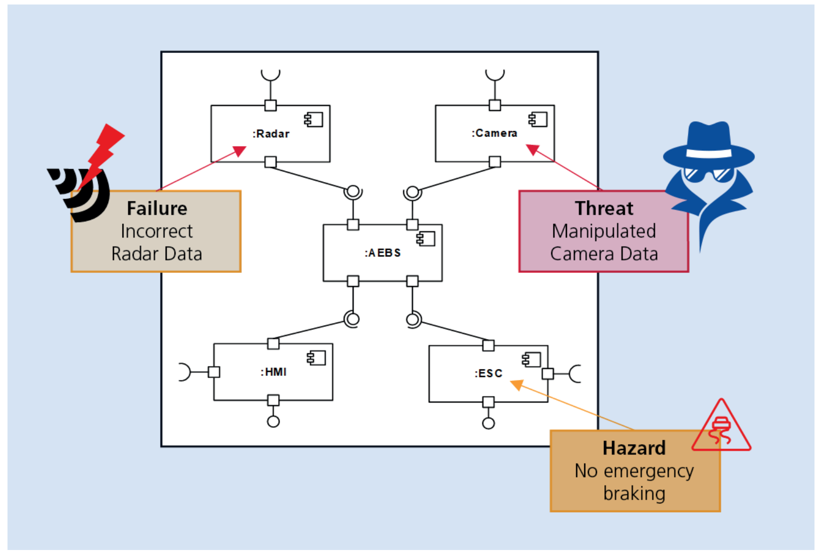


Figure 2: Component diagram of the AEBS.

Example System with HARA/TARA Results

To illustrate the method presented in this white paper, we model an exemplary system architecture of an Autonomous Emergency Braking System (AEBS) as described in UNECE R152. We base this exemplary architecture on the reference architecture of a driver assistance system from the ISO 4804 standard, which describes steps for developing and validating automated driving systems based on safety and security principles.

System Description

Based on these two documents, we assume that the system looks like the component diagram shown in Figure 2.
The figure shows a component diagram that consists of five software components. The first two components (top left and top right) represent radar and camera systems, respectively, that are used to allow the vehicle to detect obstacles while driving. Sensor inputs from these two components are fed into the AEBS component located in the center of the diagram. The AEBS calculates a brake decision based on these sensor inputs and, if a brake decision is made, sends corresponding messages to the human-machine interface (HMI) (to display a warning) and to the electronic stability control (ESC) component (to execute the actual brake command).

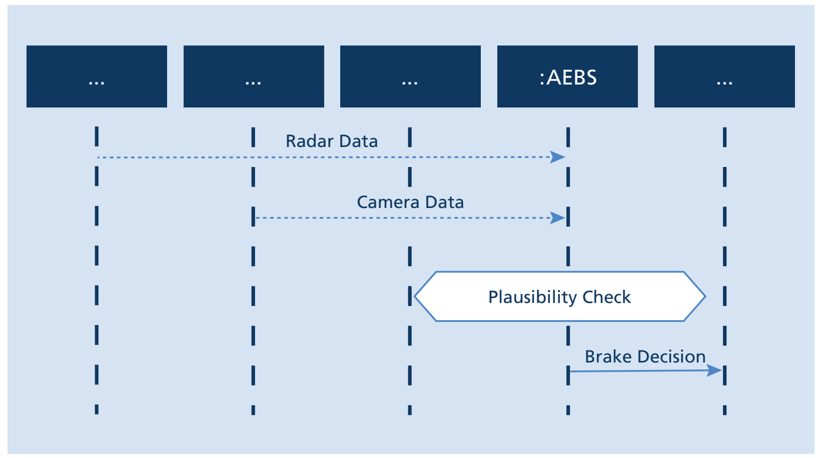


Figure 3: Dynamic Behavior of the AEBS modeled as an MSD.

In addition to this purely static architectural description of the system, we model its dynamic behavior as modal sequence diagrams (MSDs) [Har2008].
MSDs are specialized UML sequence diagrams that can be used to express requirements on the communication messages, i.e., which messages must be sent or received by the components specified in the static architecture. Modeling the dynamic behavior of the system is crucial to analyze how messages (and therefore failures related to these messages, e.g., due to an omitted message) propagate through the system. Figure 3 shows an exemplary MSD.
It consists of two messages RadarData and CameraData being sent to the AEBS component. In the MSD syntax, these two messages can be represented by dashed blue arrows to indicate that a message may be received by a component.The messages are then processed by the AEBS and checked against each other for plausibility (cf. the PlausibilityCheck fragment on the AEBS lifeline). If the plausibility check is successful, a BrakeDecision message must be sent by the AEBS. In the MSD syntax, this requirement is expressed by a solid blue arrow.

>>

HARA/ TARA

Hazard analysis and risk assessment (HARA) is a safety activity required by ISO 26262 to identify hazards, and threat analysis and risk assessment (TARA) is the corresponding security activity required by ISO 21434 to identify threats. Safety and security mechanisms are chosen based on the HARA and TARA results, and their effectiveness has to be confirmed through verification and validation.

HARA / TARA Results

Hazards and threats identified during the HARA, or TARA respectively, can be annotated to the elements in the modeled system architecture to indicate where they occur. Figure 2 shows that the threat „manipulated camera data" identified in the TARA is annotated to the camera component. In the same way, the hazard „no emergency braking" is annotated to the ESC component to express that missing emergency braking can lead to an accident and thus to injury to passengers. The dependency analysis can then be used to calculate whether and in what combination with other attacks or failures (e.g., wrong values sent from the radar component, represented by the failure "incorrect radar data" in Figure 2) the manipulation attack must occur to lead to a hazard. A test engineer should not know the detailed design specification of the system under test. Hence, when our method is used by a test engineer without inputs from system engineers, they have to make assumptions about the system internal details or its actual implementation. We demonstrate that our method can be used to derive test cases from requirements, which we derived from the generic reference architecture of the ISO 4804 without knowledge about implementation details of the system.

Safety and Security Dependency Analysis

With the help of our safety and security dependency analysis that was developed as preliminary work, we are able to automatically calculate relations between hazards and threats. Specifically, the analysis results contain information on how failures and attacks propagate in the system and eventually lead to hazards. To be able to calculate and analyze such a failure and attack propagation, the system's static architecture and dynamic communication behavior must be modeled first. For this step, existing artifacts from the requirements engineering process and early system design (e.g., UML component diagrams or other architectural descriptions as presented above) can be re-used. Vice versa, if artifacts are created exclusively for this step, they can be re-used in subsequent development process steps (e.g., to refine the system architecture). Engineers can create a system model directly in our tool by using an advanced editor that is able to process extended UML syntax.

Calculating Failure and Attack Propagation

Using the modeled static architecture (as a UML component diagram) and its dynamic behavior model (as MSDs), our tool is able to automatically calculate a failure propagation model and map identified threats to actual attack locations in it. The failure propagation model is represented by a security-enhanced component fault tree (SeCFT) [Ste2016], a special kind of fault tree analysis model on component level taking security attacks into consideration. Figure 4 shows an exemplary SeCFT for our sample system. The rectangles in the model represent its different component fault trees (CFTs). Each component in the input model is transformed into a corresponding CFT. For example, the Radar component shown in Figure 2 maps to the Radar_CFT in Figure 4. Each CFT contains one or more so-called failure modes for different kinds of failures. Failure modes are denoted as triangles on the borders of each CFT. We differentiate between omission (i.e., a message is not received), commission (i.e., a message is received unexpectedly) and value (i.e., a message is received with wrong values) failures. In our sample CFT, a failure of the radar sensor may lead to wrong sensor values being read by the Radar component. This is represented by an incoming value (V) failure of the Radar_CFT. On the other hand, an attacker may also be able to manipulate the camera vision, leading to value failures of the inputs of the Camera component. Based on the system behavior modeled in the MSDs, our tool is then able to calculate how these failures propagate through the system and eventually end up in the ESC component. If the combination of failures may result in the brake message not being sent by the ESC, this might lead to the "No emergency braking" hazard introduced in Figure 2.

Determining Critical Paths and Minimal Cut Sets

Based on SeCFTs, critical paths of failures (i.e., failures and attacks that may lead to a hazard) can be determined automatically. Critical paths are all paths of connected failure modes that eventually end up in a hazard. In addition, to see which attacks and failures would occur in combination, we are also able to calculate combinations of failures that would have to occur simultaneously to cause a hazard. This information is represented by so-called minimal cut sets (MCSs). Thereby, MCSs represent the identified safety and security dependencies. Figure 5 shows an MCS for our sample SeCFT. Based on how the failure modes in the SeCFT are related, our tool calculated that a tampering with camera data has to occur in combination with a failure of the radar sensor (and the radar sensor transmitting wrong values as a result) to lead to the hazard of an omission of emergency braking signals.

Deriving Test Cases

The minimal cut set for our sample system suggests that tampering with camera data alone will not lead to a failing emergency braking but must occur in combination with a value failure of radar data. However, to be sure that this is indeed the case, it is necessary to test this assumption. Based on the MCSs, we can therefore generate test cases automatically, to check whether the included failures can actually lead to a hazard only in combination. For the test case description, we chose a CSV representation. CSV files offer the advantage that they are used as a common exchange format that can also be inspected by humans as an intermediate step. Thereby, the intermediate test cases can be customized and adjusted before they are fed into the XIL tool chain. In the generated test cases, we can further reference the other artifacts from the safety and security dependency analysis process, e.g., to refer to messages

>>

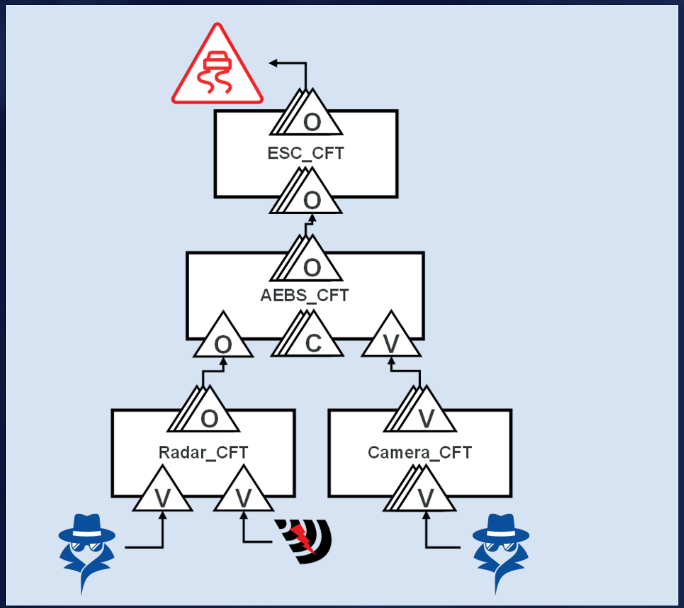


Figure 4: Sketch of the security-informed failure propagation model specified as SeCFT.

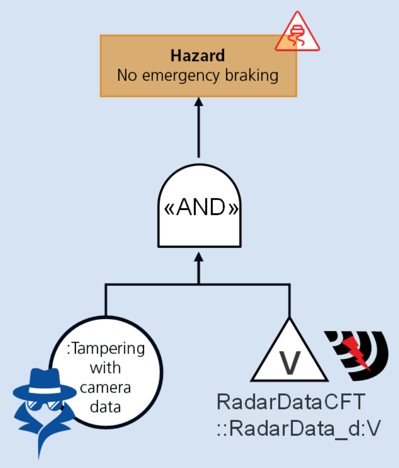


Figure 5: Minimal Cut Set (MCS) calculated from the failure propagation model shown in Figure 4.

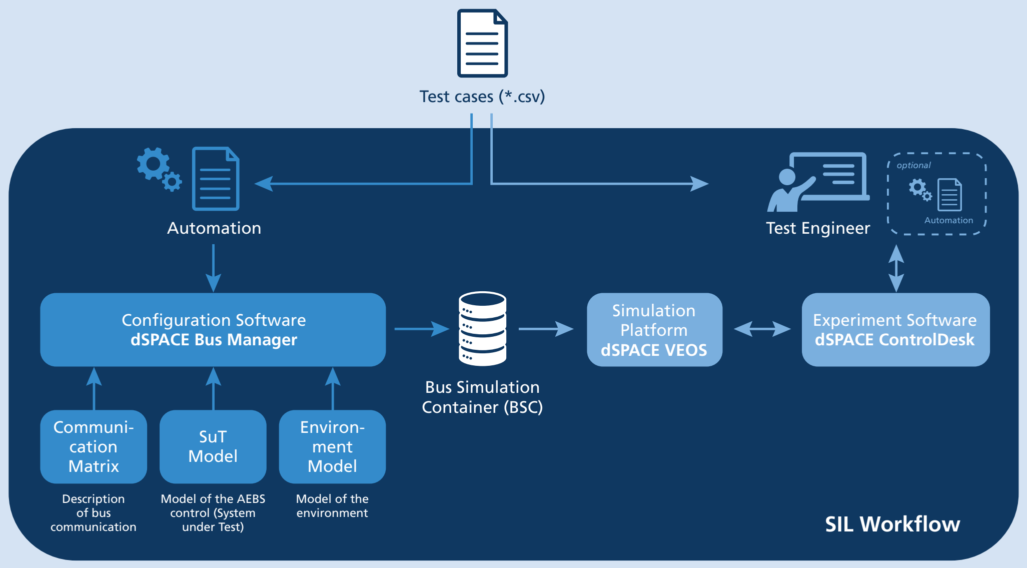


Figure 6: Schematic overview of the software-in-the-loop testing workflow. Starting from the test case description, an automation script creates a suitable configuration, including all necessary bus signals and the environment model.

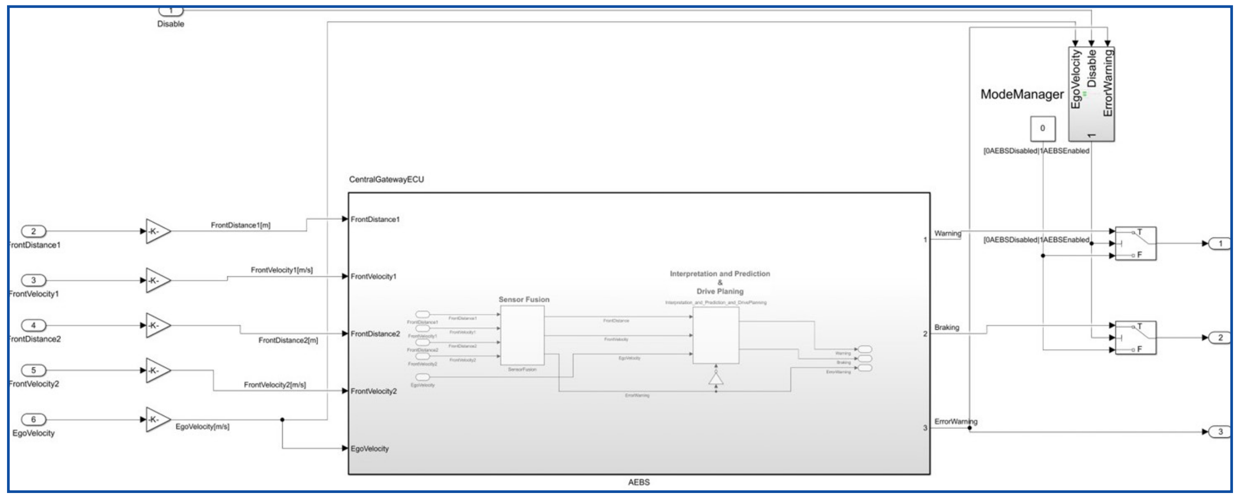
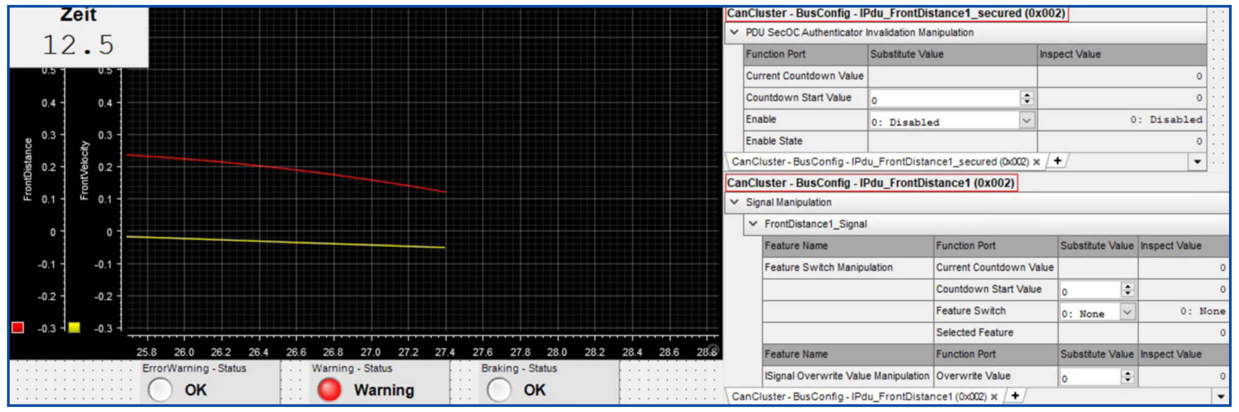


Figure 7: Schematic overview of the system under test realized as a Simulink model. The different building blocks are visible which are subcomponents of the AEBS

in the MSDs. By doing this, we can indicate which messages have to be sent "normally" and which have to be left out or sent with a more specific invalid/different value.

XIL Configuration and Execution

Hardware-in-the-loop (HIL) systems are widely used as test platforms for electronic control units (ECU). Especially in the automotive industry where a single vehicle contains dozens of ECUs, HIL test systems have become an important part of the development cycle, reducing the need for expensive real-world test drives and increasing the test coverage in general. A HIL system typically consists of real-time hardware and simulation software which together provide a realistic environment for the connected real ECU. HIL tests are reproducible and fully automatable, allowing 24/7 operation to reduce validation times.With modern vehicles becoming more software-defined, it is crucial to test and validate software as early as possible. Software-in-the-loop (SIL) testing allows the user to test software functionalities without any ECU hardware. Systems under test are virtual ECUs (V-ECUs) which come in different levels [VECU]. V-ECUs can range from consisting of just a functional model (level 0 V-ECU) up to consisting of the final binary file meant to run on the target hardware (level 4 V-ECU). A SIL system typically consists of a simulation and integration platform, the environment simulation models, the system under test, and a control software. Additionally, a supported SIL-HIL continuity in the tool chain allows the reuse of tests in both SIL and HIL testing scenarios. In case of our exemplary system – the AEBS control software prototype – we will use a SIL environment to execute the test cases resulting from the safety and security dependency analysis. The starting point is the model of the AEBS functionality, including the complete description of the bus communication. Accordingly, a suitable test environment is a SIL test platform which integrates the model of the AEBS control, i.e., the system under test (SUT), the description of the bus communication, and the model of the environment into an interactive simulation.



CanCluster - BusConfig - IPdu_FrontDistance1_secured (0x002)

PDU SecOC Authenticator Invalidation Manipulation

Function Port	Substitute Value	Inspect Value
Current Countdown Value		0
Countdown Start Value	0	0
Enable	0: Disabled	0: Disabled
Enable State		0

CanCluster - BusConfig - IPdu_FrontDistance1_secured (0x002)

CanCluster - BusConfig - IPdu_FrontDistance1 (0x002)

Signal Manipulation

FrontDistance1_Signal

Feature Name	Function Port	Substitute Value	Inspect Value
Feature Switch Manipulation	Current Countdown Value		0
	Countdown Start Value	0	0
	Feature Switch	0: None	0: None
	Selected Feature		0
Feature Name	Function Port	Substitute Value	Inspect Value
ISignal Overwrite Value Manipulation	Overwrite Value	0	0

CanCluster - BusConfig - IPdu_FrontDistance1 (0x002)

Figure 8: Screenshot of an exemplary ControlDesk layout showing a time plotter which tracks signals over time, control LEDs which track the response behavior of the AEBS system under test, and manipulation options to simulate failures and attacks.

During the simulation, the test engineer can access all variables from both the SUT model and the environment model for monitoring or manipulation purposes through an interactive experiment software. Finally, the test engineer can automatically execute all the different test cases. In the next section, we go through the consecutive steps in the tools in more detail.

Automated Test Case Configuration

The goal of the SIL tool chain is to enable the test engineer to run all test cases in a suitable environment, highly automated and without any additional manual coding. The test cases resulting from the safety and security dependency analysis are described in a .csv file – a common exchange format for tabular data. The .csv file serves as one of the inputs for the SIL testing workflow which is depicted in Figure 6. Additionally, the files of the function under development, suitable environment model files, and a communication description file are required. For our running example introduced in "Example System with HARA/TARA Results", the relevant file for the function under development is a model of the AEBS control which implements the different building blocks of the reference architecture of ISO 4804: sensor fusion, interpretation and prediction, mode manager, drive planning. See Figure 7 for a schematic overview. For the test engineer, the system under test is typically a grey box, so a functional description is known but there are no details on the actual implementation. The communication matrix file is a formal description of the communication between different ECUs within the vehicle. Depending on its usage it comes in different "cuts", e.g., description of a single ECU, an ECU subnetwork, or the entire vehicle network.
The communication matrix contains information such as signal names, identifiers, length, initial values, cycle times, authentication and encryption mechanisms, and more. For the configuration of the test cases, the communication matrix serves as a database from which signals are selected to be simulated normally and to be manipulated in the actual test execution. Optionally, a mapping file can be introduced to map the signal names used in the safety and security dependency analysis onto the actual bus and network signals described in the communication matrix of the system. Finally, a model of the environment is needed which provides the inputs of the AEBS functionality, i.e., the sensor data which belongs to specific predefined test scenarios. These scenarios can be activated during the simulation. Additionally, the environment model contains the necessary receivers of the output of the AEBS control, i.e., the warning signal to the human-machine interface and the braking signal to the brake ECU.The automation interface of the implementation tool Bus Manager allows an automatic configuration per test case. This includes the configuration of all involved signals, i.e., the signals which are received and transmitted by the SUT, as well as features for manipulation and monitoring purposes. The different features allow the simulation of all the possible failures and attacks which are in scope of the HARA and TARA and correspondingly are part of the test cases resulting from the safety and security dependency analysis. Finally, a container is created consisting of the bus and network communication configuration, the environ-

>>

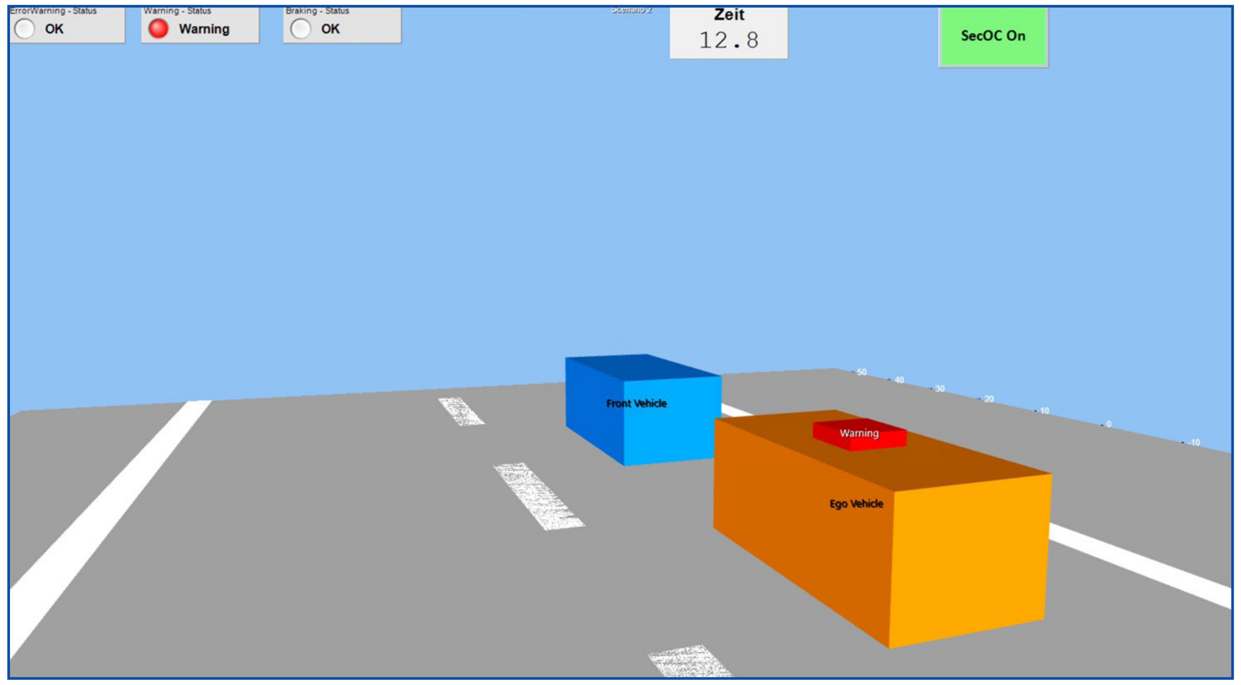


Figure 9: Simple 3D animation within ControlDesk to visualize a specific driving scenario providing more context to each test case.

ment model, and the model of the SUT. For our example, we use a python automation script which reads out the .csv file and then per test case does the following: import all relevant files in the Bus Manager tool, select the involved signals, activate the necessary features, and create the final container.

Automated Test Case Execution

For the actual test case execution, the PC-based simulation platform VEOS is used which enables software-in-the-loop testing during development of electronic control units. VEOS integrates the different artefacts (V-ECU under test, environment model, bus configuration) into one executable simulation application.The interactive experiment software ControlDesk allows access to all variables at simulation run time and hence, enables the test engineer to perform automated and manual testing. Customizable layout options help to control and visualize the simulation. For example, Figure 8 shows a simple layout which tracks the distance and velocity over time of the vehicle in the front and monitors the response behavior of the system under test, i.e., brake request, collision warning, and error warning, as well as manipulation options of different signals to simulate failures and attacks. Additionally, Figure 9 shows a simple 3D animation within ControlDesk of a specific driving scenario providing more context to each test case. In our example, the final stage is manual test execution in ControlDesk. This stage could be advanced even further by leveraging capabilities of AutomationDesk for a completely automated test execution.

Summary and Outlook

In this whitepaper, we presented a tool-supported method to automatically derive test cases for identified safety and security dependencies. Our method is aligned with existing safety and security standards of the automotive domain and is designed to enable the reuse of existing artifacts from these processes. To improve the overall efficiency, the generated test cases can be imported into the existing dSPACE testing tool chain to further reduce time-consuming manual work and the susceptibility to errors. We evaluated our method on an example of an automated emergency braking system based on the ISO4804 standard and have developed an end-to-end demonstrator. Our method thereby paves the way towards an integrated safety and security-by-design life cycle in the automotive domain. An automated approach further helps engineers to keep pace with developments in the field of software-defined vehicles and to achieve safety and security goals of modern vehicles. The increased use of communication technologies such as Automotive Ethernet has influences on the interplay of attacks and failures and places new demands on countermeasures. Upcoming research projects are planned to build on the presented results and further examine the influence of future technologies on the intersection of safety and security.

About Fraunhofer IEM
What does the future of engineering look like? Fraunhofer IEM in Paderborn works on convincing solutions to this question, covering the business idea, implementation, and market success stages in the process. Through it all, the focus is on intelligent products, production systems, services, and software applications. Scientists take an interdisciplinary approach to working on new methods, tools, and processes – and use innovative technologies to ensure the long-term competitiveness of customers and partners.

Further info at *www.iem.fraunhofer.de*

Acknowledgements

This research was funded by the Ministry of Economic Affairs, Industry, Climate Action, and Energy of the State of North Rhine-Westphalia (MWIKE) in the context of the Leading-Edge Cluster 'Intelligent Technical Systems OstWestfalenLippe (it's OWL)' and supervised by Project Management Jülich (PtJ). The responsibility for the content of this publication lies with the author.

Authors

Fraunhofer Institute for Mechatronic Systems Design IEM
Dr. Markus Fockel
markus.fockel@iem.fraunhofer.de

Roman Trentinaglia
roman.trentinaglia@iem.fraunhofer.de

dSPACE GmbH
Dr. Matthias Pukrop
mpukrop@dspace.de

Tobias Schaeffer
tschaeffer@dspace.de

Literature

[Ste2016] Steiner, M. (2016). Integrating Security Concerns into Safety Analysis of Embedded Systems Using Component Fault Trees. PhD thesis, TU Kaiserslautern.

[Upstream2024] Upstream Security: 2024 Global AutomotiveCyber security Report, Network Security, Volume 2024, Issue 1.

[Foc2022] Fockel, M., Schubert, D., Trentinaglia, R., Schulz, H., & Kirmair, W. (2022). Semi-automatic Integrated Safety and Security Analysis for Automotive Systems. In MODELSWARD 2022.

[Har2008] Harel, D. & Maoz, S. (2008). Assert and negate revisited: Modal semantics for UML sequence diagrams. Software & Systems Modeling, 7(2):237–252.

[VECU]
Prostep IVIP: White Paper - SmartSE - Virtual Electronic Control Units, 2020.

05/2024