Model-based Security and Safety Assurance for Automotive Safety Systems

Extended Abstract

Sanjana Biank
sanjana.biank@carissma.eu
Technische Hochschule
Ingolstadt/Research Center
CARISSMA

Matthias Meyer
matthias.meyer@carissma.eu
Technische Hochschule
Ingolstadt/Research Center
CARISSMA

Thomas Hempen
thomas.hempen@carissma.eu
Technische Hochschule
Ingolstadt/Research Center
CARISSMA

Werner Huber
werner.huber@thi.de
Technische Hochschule
Ingolstadt/Research Center
CARISSMA

Hans-Joachim Hof
hof@thi.de
Technische Hochschule
Ingolstadt/Research Center
CARISSMA

ABSTRACT

New functions of modern vehicles (e.g., autonomous driving, early airbag ignition) make heavy use of internal and external communication. The increased usage of communication for the realization of safety-critical functions leads to new challenges for security and safety. In order to meet current as well as future requirements regarding the validity of autonomous vehicles, a holistic (regarding security and safety), systematic and traceable assurance methodology is required. In this extended abstract an approach for a model-based assurance methodology for both security and safety is introduced.

CCS CONCEPTS

• **Security and privacy → Formal security models**; *Distributed systems security*; • **Software and its engineering →** *Software verification and validation.*

KEYWORDS

Model-based Testing, Security, Safety, Verification, Validation

ACM Reference Format:
Sanjana Biank, Matthias Meyer, Thomas Hempen, Werner Huber, and Hans-Joachim Hof. 2019. Model-based Security and Safety Assurance for Automotive Safety Systems: Extended Abstract. In *Proceedings of 3. ACM Computer Science in Cars Symposium (CSCS 2019).* ACM, New York, NY, USA, 3 pages.

1 MOTIVATION

An increasing number of automotive safety systems that perceive the driving environment and autonomously react in case of emergency is being launched. The trend is towards fully autonomous vehicles, which are expected to master all possible situations without intervention of the driver. The high interconnectedness of such vehicles with their environment as well as the increased intra-vehicle-communication, give rise to new challenges due to intentional, malicious manipulation of vehicle data. Miller and Valasek demonstrated an attack on a Chrysler Jeep. They exploited a security breach in the infotainment system, which enabled them to remotely control the Jeep's breaks and gas - amongst other things [2]. This shows the need of security for safety-critical functions. Current security legislation in California [1] indicates that future vehicle generations will need security certification. Security can neither be considered on its own, functional safety, safety of the intended functionality and cybersecurity have to be considered in a holistic approach. In order to meet current as well as future requirements regarding the validity of autonomous vehicles, a holistic (regarding safety and security), systematic, reproducible and traceable assurance methodology is required. This extended abstract introduces a model-based approach to such a systematic and traceable methodology for security and safety.

2 MODEL-BASED APPROACH TO SAFETY AND SECURITY ASSURANCE

Model-based methods will be used for both security and safety. Early in the development process, parallel to the development activities, models for both security and safety will be constructed for testing purposes. Building models aids in detecting missing or flawed requirements. Linking test information in the models to requirements allows tracing the connection between requirements and the associated use cases. The models will be the basis for an automated and systematic test case generation. More information on model-based testing can be found in [9, 10]. The test cases will be executed in a simulation environment in order to perform tests as early as possible. Both security and safety will be part of this methodology, but each has a different focus. An intensified and more systematic view on security will be achieved through the consistent use of model-based methods. Modeling and test generation for complex, interconnected and parallel functions will be addressed in the safety part. Synergies between security and safety as well as possible contact points need to be identified in order to examine interactions between security and safety.

CSCS 2019, Oktober, 2019, Kaiserslautern, Germany

Modeling of security is twofold in our approach. It includes attacker modeling as well as attack modeling. An attacker model describes the abilities as well as the goals of an attacker. See [7] for details. This information is used to determine the attack surface of the system at test. Also, the goal of an attacker is an important input for the attack model. The attack model describes a strategy to achieve the goal of an attacker. The strategy consists of several phases. Each phase is described by a model of the activities in this phase. Parts of the activities are attacks that may be modeled using attack trees. Leafs of an attack tree can be used to generate test cases. Such a layered approach to security modeling allows heavy reuse of models, e.g., leafs of the attack tree.

Regarding safety, two main goals are pursued: To be able to execute test cases as early as possible in a virtual simulation and to be able to model and generate test cases for complex, interconnected and parallel automotive functions. The test model and the test case generation therefore need to deal with concurrency, dependencies on different internal and external events as well as test information for the simulation environment. UML2 provides means to model concurrency using fork/join for activity diagrams and state diagrams as well as orthogonal states with concurrent regions for state diagrams. This will be explored as well as the possibility to describe some aspects in separate models. While it is possible to model concurrency using UML, to the best knowledge of the authors there is no tool available to generate test cases from UML models featuring concurrency. In order to use the test information in the test model to automatically generate test cases, the graphical representation which is understandable by the user, needs to be transformed in to a graph representation on data level. The information, which may be contained in different models needs to be integrated into a single graph. Concurrencies need to be resolved by considering possible execution orders of the test steps (A before B, B before A, and A and B happening at once). Moreover the algorithm needs to consider guard conditions to only form valid test paths. It has to be taken into consideration how to deal with test information for the simulation environment when building the data model. The obtained graph representation is used to generate test cases based on search algorithms. In order to deal with large models, memory space and computing power need to be taken into account.

3 RELATED WORK

Functional safety tests are standardized in the ISO 26262 "Road Vehicles-Functional Safety", but security test are not yet standardized. Solely the SAE J3061 "Cybersecurity Guidebook for Cyber-Physical Systems" does describe methods and processes for the development of secure systems inspired by the ISO 26262 as well as differences and synergies between both [8]. The future standard ISO/SAE CD 21434 "Road Vehicles-Cybersecurity Engineering" is under development [5]. Kriebel et al. introduce a methodology for model-based test case creation in automotive software engineering [6]. Test cases are generated on the basis of a functional model implemented as an Activity Diagram (AD) in SysML. The AD is part of a BMW-specific semi-formal specification format for requirements, design and test. This work is interesting because it considers generating test cases from ADs with concurrency. The concurrencies are resolved by transforming parallelism into a sequential control flow in a defined order. As generating test cases is not the main subject of this paper no further details are provided. Guan et al. use three different graphs to model the relationship between components, the dependencies on interfaces and states in components as well as concurrent events [3]. They combine state diagrams and sequence diagrams and use an experimental tool of their own to build the models and generate the test cases. However not all paths generated by traversing the model are necessarily feasible. When there are guard conditions the paths must be inspected manually. Hempen et al. generate test scenarios for a simulation environment using MBT [4]. They use UML state diagrams to model the different systems states and possible transitions between them and added test information for the simulation environment through tagged values. Concurrency was not addressed. The test case generation is automated but test cases may still have to be adapted.

4 CONCLUSION

The presented approach uses model-based methods for security and safety assurance. While it is only a concept and there are still questions regarding the practical implementation and the interaction of security and safety in the methodology, it is the belief of the authors that a systematic and traceable assurance methodology for both security and safety is essential for automated driving functions.

ACKNOWLEDGMENTS

Federal Ministry of Education and Research

This work is supported under the KMU-innovativ program of the German Federal Ministry of Education and Research (BMBF) under Grant No. 16KIS0946.

REFERENCES

[1] Suhasini Gadam. 2018. Why California Has The Better Set Of Regulations For Autonomous Vehicles? Retrieved August 20, 2019 from https://medium.com/datadriveninvestor/why-california-has-the-better-set-of-regulations-for-autonomous-vehicles-e45411384531

[2] Andy Greenberg. 2015. Hackers Remotely Kill a Jeep on the Highway- With Me in It. Retrieved August 21, 2019 from https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/?mbid=social_twitter

[3] Jing Guan and Jeff Offutt. 2015. A model-based testing technique for component-based real-time embedded systems. In *IEEE Eighth International Conference on Software Testing, Verification and Validation workshops (ICSTW), 2015*. IEEE, Piscataway, NJ, 1–10. https://doi.org/10.1109/ICSTW.2015.7107407

[4] Thomas Hempen, Sanjana Biank, Werner Huber, and Christian Diedrich. 2018. Model Based Generation of Driving Scenarios. In *Intelligent Transport Systems – From Research and Development to the Market Uptake*, Tatiana Kováčiková, Ľuboš Buzna, Ghadir Pourhashem, Giuseppe Lugano, Yannick Cornet, and Nathalie Lugano (Eds.). Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Vol. 222. Springer International Publishing, Cham, 153–163. https://doi.org/10.1007/978-3-319-93710-6{_}17

[5] International Organization for Standardization. 2018. ISO/SAE CD 21434 Road Vehicles: Road Vehicles – Cybersecurity engineering.

[6] Stefan Kriebel, Johannes Richenhagen, Matthias Markthaler, Karin Samira Salman, Timo Greifenberg, Steffen Hillemacher, Bernhard Rumpe, Christoph Schulze, Andreas Wortmann, and Philipp Orth. 2018. Improving model-based testing in automotive software engineering. In *2018 ACM/IEEE 40th International Conference on Software Engineering: Software engineering in practice*, Frances Paulisch and Jan Bosch (Eds.). IEEE, Piscataway, NJ, 172–180. https://doi.org/10.1145/3183519.3183533

[7] Christoph Ponikwar, Hans-Joachim Hof, Smriti Gopinath, and Lars Wischhof. 2016. Beyond the Dolev-Yao Model: Realistic Application-Specific Attacker Models for Applications Using Vehicular Communication. http://arxiv.org/pdf/1607.08277v1

[8] Society of Automotive Engineers. 2016. SAE J3061: Cybersecurity Guidebook for Cyber-Physical Vehicle Systems.

[9] Mark Utting, Alexander Pretschner, and Bruno Legeard. 2012. A taxonomy of model-based testing approaches. *Software Testing, Verification and Reliability* 22, 5 (2012), 297–312. https://doi.org/10.1002/stvr.456

[10] Mario Winter, Thomas Roßner, Christian Brandes, and Helmut Goetz. 2016. *Basiswissen modellbasierter Test: Aus- und Weiterbildung zum ISTQB Foundation Level–Certified Model-Based Tester* (2., vollständig überarbeitete und aktualisierte auflage ed.). dpunkt.verlag, Heidelberg.