2021 International Conference on Computational Science and Computational Intelligence (CSCI) | 978-1-6654-5841-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/CSCI54926.2021.00179

Model-Based Security Testing of Vehicle Networks

Florian Sommer
Institute of Energy Efficient Mobility
Karlsruhe University of Applied Sciences
Karlsruhe, Germany
florian.sommer@h-ka.de

Reiner Kriesten
Institute of Energy Efficient Mobility
Karlsruhe University of Applied Sciences
Karlsruhe, Germany
reiner.kriesten@h-ka.de

Frank Kargl
Institute of Distributed Systems
Ulm University
Ulm, Germany
frank.kargl@uni-ulm.de

***Abstract*—Modern vehicles consist of a large number of electronic information technology components, which communicate with each other and external components. To protect vehicles against security attacks, automotive-specific standards and regulations require an integration of security concepts and measures in vehicles. Security testing techniques, such as penetration tests, are used to verify and validate those measures. However, these methods are usually carried out manually in late phases of development. Thus, identified vulnerabilities can only be eliminated at a late stage leading to a high investment of time and resources. This paper presents a model-based security testing approach which aims to enable security tests early on in the vehicle development process in an automated way. This allows vulnerabilities to be identified and eliminated at an early stage during development. Therefore, we show our concept to create a security model based on a vehicle network. This model can be used to automatically derive attack paths for security testing. We further illustrate our approach by applying it to a real-world vehicle network.**

***Index Terms*—automotive security, model-based security testing, automation**

I. Introduction

The complexity of modern vehicles has grown continuously in recent years as a result of an increasing number of electronic information technology components in vehicular networks. Especially the trend towards autonomous driving leads further to an increased communication with external components and a high amount of sensor data to recognize a vehicle's environment. For this reason, the specification of security requirements becomes more important to protect modern vehicles against security attacks and unauthorized access. The feasibility of such attacks has been demonstrated in recent years by several publications as shown in our survey of automotive security attacks [1]. Thus, new standards, such as ISO 21434 [2], and regulations, such as UN R155 [3], specify activities during the vehicle development process and lifecycle in order to assure a certain level of security. These standards mainly address activities such as assessing the risk of possible attacks during a Threat Analysis and Risk Assessment (TARA) [2] or the derivation of security mechanisms. Furthermore, security testing is required as an important step within development to verify and validate implemented security measures and to identify vulnerabilities in vehicles. For this purpose, test techniques such as vulnerability scanning, fuzzing, and penetration testing are suggested by these standards.

This work was developed in the project SecForCARs-SAVE (reference number: 16KIS0796) which is funded by the German ministry of education and research (BMBF).

Problem: Vulnerability scanning, fuzzing, and penetration testing are dynamic test methods which require test objects to be fully developed. Thus, these test methods are usually employed at the end of development. As a result, a validation of implemented security measures as well as the identification of potential vulnerabilities, which were not covered by these measures, can only take place at a late stage. The ISO 21434 considers a dependency on late testing to be a problem and argues that security and its validation should already start in the concept and design phase. Furthermore, test techniques such as penetration testing are usually carried out manually. Marksteiner et al. [4] argue that manual security test methods reach their limit when it comes to the high complexity of modern vehicles. Associated time and resource constraints were considered by Kasoju et al. [5] as a main challenge in automotive testing processes.

Solution: In order to automate and enable early security testing of vehicles, we introduce a concept for model-based security testing in this paper. Our survey of security attacks on vehicles [1] has shown that such attacks often consist of paths with several attack steps across a vehicle's network. For this purpose, we consider an internal vehicle network (E/E architecture) as the system under test. Based on a suitable definition of the architecture's elements, we create a security model, which includes implemented security measures and attack-related characteristics, such as vulnerabilities. This enables the derivation of attack paths across the vehicle network for security testing. By applying model-based methods to security testing, vulnerabilities can potentially be identified early on through automated tests. In this way, existing test techniques, such as penetration testing, can be complemented.

Contribution: This paper illustrates our concept of creating a security model based on a vehicle network, its environment, and attack-related characteristics by taking an attacker's perspective. We apply this security model to a real-world vehicle network in order to evaluate our approach. Furthermore, the principal process for deriving attack paths from the model is shown. Finally, we outline the potential of our approach in order to analyze and test vehicle networks for potential vulnerabilities early and continuously during development in an automated way.

This work is structured as follows: In Section II, we give an overview of model-based (security) testing and related work. Section III presents our concept in which we create a security model. In Section IV, we show a proof of concept by applying our approach to a real-world system. In Section V we discuss challenges of our approach and potential solutions. In Section VI, a conclusion is drawn and an outlook on future work is given.

II. Background and Related Work

This section provides an overview of model-based software and security testing as well as related work.

A. Model-Based Testing

Model-based test methods are used to automate the test process and to detect errors early and more efficiently [6]. By creating formal models of a system in the concept phase of development, early testing is possible, which usually leads to cost and resource savings [6]. This can be done, for example, with Unified Modeling Language (UML) diagrams to model a system. These models are used to automatically derive test cases, which is usually done by applying a test specification to the system model. A test specification contains information on how tests are generated from a model (for example, via coverage criteria). As an example, the software of an airbag Electronic Control Unit (ECU) could be modeled as a state machine. By using path coverage, test paths can be derived to test the software implementation. Model-based test methods are applied in several domains, such as Information Technology (IT) [7], or robotics [8]. While model-based testing of software is also applied in the automotive domain [9], security-related approaches are not widely used [10].

B. Model-Based Security Testing

Model-Based Security Testing (MBST) combines aspects of security testing with model-based methods [11]. Security testing in general verifies that security mechanisms have been implemented correctly, and in particular detects vulnerabilities in a system. This is usually achieved through test methods, such as penetration testing. In general, a security tester acts similarly to a malicious attacker. Both try to identify vulnerabilities in a target system by finding and executing attacks. Thus, attacks and their description are an important factor in security testing. For this reason, MBST aims to describe a system in a formal way in order to generate security attacks automatically for test execution.

MBST is applied in different domains. As a result of the world-wide connection of IT systems through the internet, such test methods are especially employed for web applications [12] and access control systems [13]. The work of Hagerman et al. [14] demonstrates an application of model-based security testing in the aerospace domain. Brucker et al. [15] present an approach which is applied in healthcare. The work of LeMay et al. [16] shows a state-based security evaluation method which results in an executable model of an electric power Supervisory Control and Data Acquisition (SCADA) system to simulate attack behavior.

In the automotive domain, model-based security testing is still new and not yet established, since security in general has just recently become a relevant factor of automotive industry and academia. However, Cheah et al. [17] present an automotive MBST approach in which attack trees are formally described using Communicating Sequential Processes (CSP). Formal verification and model checking methods are used to derive test cases in order to test Controller Area Network (CAN) [18] and Bluetooth communication channels. Mundhenk et al. [19] present an approach for analyzing vehicular networks. By describing networks as Markov chains, risk metrics from security and safety are applied to determine probabilities of the state transitions. Approaches such as the work of Oruganti et al. [20] use Matlab/Simulink as a framework for performing security tests in a virtual environment based on Simulink vehicle models. Volkersdorfer et al. [21] introduce a model-based concept to automate security reviews and tests during the automotive software engineering process. For this purpose, an attacker model and a target model are used to simulate attacks on a target system.

III. Automotive Security Model

In this section, we present a model-based security testing approach, which aims to automate the security test process (or parts of it). This also enables early tests during vehicle development. Since performing security tests is closely related to executing an attack, it is important to understand how vehicle networks can be attacked. For this purpose, we collected published security attacks on vehicles [22] and created a taxonomy [1] to describe them sufficiently. Results suggest that attackers were generally able to gain access to vehicles through an internal interface (e.g., On-Board Diagnostics (OBD) connector or Bluetooth [23]). Subsequently, attackers were able to access and compromise various components within a vehicle network (for example, getting access to an ECU in order to control the vehicle's steering). Such attacks are thus characterized by attack paths. An attack path consists of several attack steps starting at a vehicle interface and leads through the internal network to a target component. For this reason, the vehicle's E/E architecture, implemented security measures, and characteristics, which are related to possible attack paths, are particularly relevant. The model-based approach illustrated in this work presents a way to create a security model based on these inputs. This model is used to derive attack paths automatically for security tests. Figure 1 gives an overview of our overall approach. In this paper, our focus is on creating the security model of a vehicle's E/E architecture, taking into account its environment and attack characteristics. Further steps to reach our goal of early and automated security tests, such as a formal description of the model or techniques for deriving attack paths, are briefly outlined in Section V.

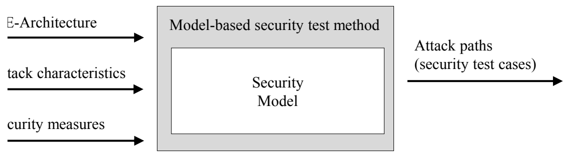
E-Architecture
tack characteristics
curity measures
Model-based security test method
Security Model
Attack paths (security test cases)

Fig. 1. The model-based security testing approach uses a vehicle's E/E architecture, attack characteristics, and security measures as inputs to create a security model. Based on that model, attack paths can be generated.

A. E/E Architecture

ISO 26262 [24] describes an E/E architecture as a system which consists of electrical and electronic elements. This comprises components, such as ECUs, sensors, and actuators, as well as their connections to each other. Wired communication systems, such as CAN or Ethernet, and wireless communication systems, such as Bluetooth or mobile communications, are taken into account. This includes all interfaces a vehicle offers to the outside world. We consider an E/E architecture to be the primary artifact describing a vehicle as a model, as it includes all elements which may be affected by a cyber attack. Furthermore, an E/E architecture is already fixed in the concept and design phase of the development process, so our model-based approach can start at an early stage.

B. Security Measures and Further Development Artifacts

Security measures aim to protect vehicles against attacks. Examples include data encryption, authenticated communication via Secure Onboard Communication (SecOC) [25], or access control on an ECU [26]. Such measures especially reduce the probability to exploit vulnerabilities and make attacks more difficult for an attacker. Thus, security mechanisms implemented on components or communication systems also have to be considered in our security model. The implemented mechanisms depend on which system elements have been assessed as critical by the manufacturer or supplier during the Threat Analysis and Risk Assessment (TARA). Currently, we only take security mechanisms into account. However, for future work, a connection between our model-based security testing approach and TARA can be created by taking additional artifacts, such as threats and their risk, into account in order to prioritize critical areas of the system.

C. Attack Characteristics

In order to describe security-related properties in our model, we analyzed our attack collection [22] to identify characteristics which are essential for describing attacks and attack paths within an E/E architecture. We especially found that the violated security property as well as the exploited vulnerability are relevant for describing an individual attack. The exploited vulnerability is basically the reason an attack is possible in the first place, while the violated security property describes the effect of an attack to a system. However, in order to describe attack paths, which involve several elements of a vehicle's E/E architecture, additional information is necessary. For this purpose, we introduced the principle of *Attacker Privileges* in [27]. These privileges describe abstract states on a component or communication system an attacker can gain by executing an attack. Thus, they are not viewed from classical sense of an access control or privilege management, but from an attacker's point of view. Five privileges are distinguished:

- **Read/Write (Functional Communication Link)**: describes an ability to read and write data on a communication system. For communication channels, which are not protected by security measures, an attacker achieves this ability as soon as an access to this channel exists. If security measures, such as encryption, are implemented on a channel, an attacker will gain that privilege only when data can be interpreted (i.e., an attacker can encrypt and decrypt data in this case).
- **Execute (Functional Component)**: describes the ability to execute functions which are implemented on a component, for example, by triggering diagnostic services on an ECU.
- **Read (Functional Component)**: describes an attacker's ability to read or extract data from a component (for example, extracting firmware of an ECU, or reading sensitive information).
- **Write (Functional Component)**: enables an attacker to write data on a component. For example, a buffer overflow attack could be performed which allows an attacker to change data on an ECU.
- **Full Control (Functional Component)**: describes that an attacker has completely taken over a component. This privilege is comparable to root privileges on an IT system and can be achieved, for example, by flashing a manipulated firmware update.

For a more detailed overview of how attackers were able to obtain certain privileges, please refer to [22]. In general, there is no hierarchy between the *Attacker Privileges* (although it could be assumed that the *Full Control* privilege includes the other privileges). However, it can be assumed that a direct internal access to communication interfaces of a component is possible as soon as an attacker can reach the *Write* or *Full Control (Functional Component)* privilege. This is a key aspect for our approach in order to model attack paths, since an attacker needs to reach one of these two privileges in order to access further attached communication systems and components. In this way, attack paths through an E/E architecture can be described as a chain of *Attacker Privileges*, which can be achieved by performing a set of attacks or attack steps. In [27], we illustrated how this approach can be used to automatically generate attack trees during a Threat Analysis and Risk Assessment (TARA). In the following chapter, we outline how we use that approach to create a security model.

D. Creating the Security Model

The basis for creating a security model is a vehicle's E/E architecture as described in Section III-A and its security mechanisms, which are implemented on elements of the E/E architecture. Additionally, the environmental entities, which are affecting a vehicle's security, have to be considered.

These entities generally consists of all information technology elements with an influence on a vehicle. Hence, we include every external entity which communicates with the vehicle by one of its interfaces. This could be, for example, a backend server or smartphone. Entities with an influence on vehicle sensors, such as cameras or distance sensors, are also taken into account. Finally, the attacker or tester is a part of the environment. In the upper part of Figure 2, an overview of this concept is illustrated.

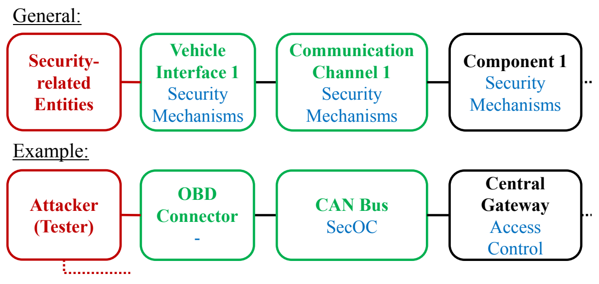

Fig. 2. Subset of an E/E architecture including security-related external entities and implemented security measures.

The bottom part of Figure 2 shows an example of our concept in which an attacker (tester) is connected to the OBD interface of the vehicle. OBD is attached to a Central Gateway by a CAN Bus. In this example, the Central Gateway is secured by an Access Control, while the CAN Bus uses an authenticated communication via SecOC. So far, our model consists of an E/E architecture and its external entities as well as implemented security measures. In the next step, we apply the *Attacker Privileges* presented in Section III-C. Thus, each communication system can exclusively take the privilege *Read/Write (Functional Communication Link)* and each component can exclusively take the privileges *Execute, Read, Write, or Full Control (Functional Component)*. In order to obtain a certain privilege, an attacker has to execute an attack which exploits a specific vulnerability. For this reason, vulnerabilities are also included into the model, which is illustrated in Figure 3.

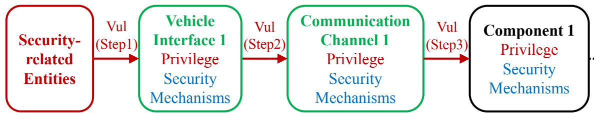

Fig. 3. Security model with integrated *Attacker Privileges* and vulnerabilities.

Each model element now contains *Attacker Privileges*, which can be achieved by exploiting a vulnerability. Thus, we define a vulnerability on an abstract level as a privilege which is achieved as well as a security property which is violated. For this purpose, we use the properties confidentiality, integrity, availability, authentication, authorization, and non-repudiation. Thus, a vulnerability always consists of one of these six security properties in combination with one of the *Attacker Privileges*. To illustrate our modeling concept, a concrete example of a security model is given in Figure 4.

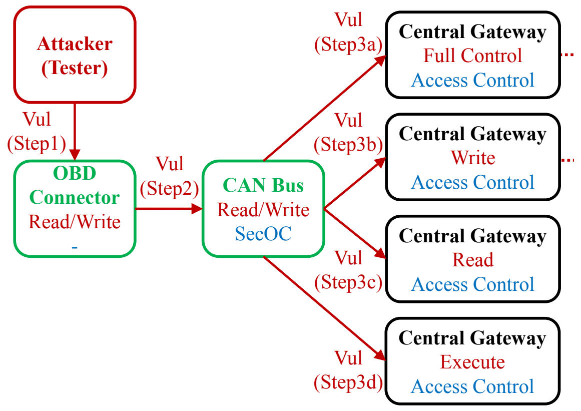

Fig. 4. Example of a security model with integrated privileges and vulnerabilities (Vul). For space reasons, transitions between states of the Central Gateway have been omitted.

In this example, an attacker (tester) is connected to CAN and a Central Gateway by an OBD Connector. We treat the OBD Connector as a communication medium, since it is merely a physical connector using CAN for communication. CAN and the Central Gateway have privileges as described in Section III-C. To achieve *Read/Write* on CAN, an attacker must exploit a vulnerability to break or bypass SecOC to communicate with the Central Gateway. To gain a specific privilege on the gateway, a vulnerability, which leads to that privilege, must be exploited. Switching between privileges within the gateway has not been illustrated in Figure 4 for conciseness reasons. When *Write* or *Full Control* privilege is reached, an attacker has the option of addressing further communication systems in order to access further components of the E/E architecture. This allows attack paths to be modeled across an E/E architecture. If an attacker is unable to exploit a vulnerability within such a path, no further privileges can be acquired. However, attacks based on already obtained privileges are still possible.

This section presented our basic modeling process to create a security model. Currently, we are working on a formalization of the model as an Extended Finite State Machine (EFSM) in which the model elements consisting of privileges are treated as states and the vulnerabilities as transitions between the states. A formal description enables us to automatically generate our security model as an EFSM by providing a vehicle's E/E architecture and its implemented security measures. In [27], we have already shown the feasibility of a formal model of our approach for an automated attack tree generation for TARA.

IV. Proof of Concept

To evaluate our concept of creating security models, we apply the approach to a real-world vehicle network. We further show that the resulting model includes attack paths corresponding to attacks which were successfully executed by security researchers in the past. For this purpose, we use the E/E architecture of a vehicle which was attacked by the researchers of Keen Security Lab in their security test report [28]. A subset of this architecture is shown in Figure 5.

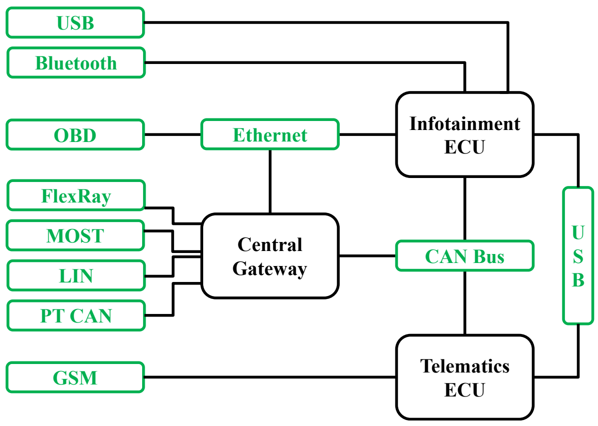

Fig. 5. E/E architecture subset of cars Keen Security Lab attacked in [28]. Green elements illustrate communication channels and interfaces.

The architecture includes a Central Gateway, an Infotainment ECU, and a Telematics ECU. All three ECUs are connected by a CAN Bus. The Infotainment and Telematics ECU also communicate via an Universal Serial Bus (USB) connection. The Central Gateway and Infotainment ECU are connected to an OBD interface by an Ethernet channel. Each ECU has additional communication interfaces. The Central Gateway is connected to the vehicle's internal network by several bus systems. The Telematics unit communicates with outside components via a Global System for Mobile Communications (GSM) connection. The Infotainment unit has a Bluetooth and USB interface. In principle, an attacker can connect to any interface to carry out attacks. Keen Security Lab researchers also demonstrated this in their report [28] by choosing different entry points and as a result different attack paths. We applied our modeling concept (see Section III) to the architecture. The resulting security model is shown in an abbreviated form in Figure 6. For space reasons, this model only shows the Telematics ECU, which communicates with the outside via GSM and with the Infotainment ECU via USB. The latter also communicates with CAN. The Central Gateway, its interfaces, and all other interfaces of Infotainment and Telematics are not included in Figure 6. The depicted model shows an attacker trying to communicate with the Telematics unit via GSM in order to get further access to ECUs of the network. For this purpose, a vulnerability (Vul(Step1)) has to be exploited on the GSM channel first, which leads to the *Read/Write* privilege. In this case, GSM uses encryption and signature algorithms to secure communication, which make an attack more complicated. However, if an attack is successful, the attacker can try to reach a certain privilege by exploiting Vul(Step2a) - Vul(Step2d) on the Telematics unit. If *Write* or *Full Control* privilege is obtained, access to the USB interface will be possible via Vul(Step3a) or Vul(Step3b). In a next step, the attacker could try to get access to the Infotainment unit by exploiting Vul(Step4a) - Vul(Step4d). In this example, the firmware of the Infotainment unit is secured by a signature algorithm, so acquiring the *Write* or *Full Control* privilege is complicated for the attacker. However, if he finds a vulnerability which leads to one of these privileges, further communication with the CAN bus is possible. Overall, our security model is used to represent security-related characteristics of a vehicular network. Thus, we can analyze that model for attack paths, which can be used for security testing.

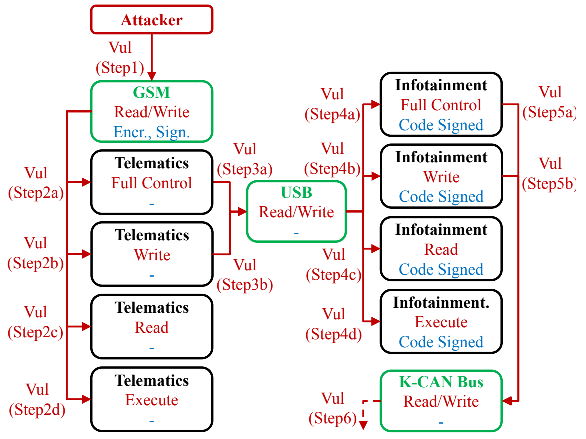

Fig. 6. Security Model based on the E/E architecture from Figure 5. Only the GSM connection to the Telematics unit and its USB connection to the Infotainment ECU are illustrated due to conciseness.

To evaluate whether our modeling approach is able to represent realistic attack paths, we analyzed the model from Figure 6 and compared it to the attacks carried out by the Keen Security Lab researchers on the vehicle network our model is based on (see Figure 5). We found that our model contains several attack paths that were executed on the real vehicle. The following path will be examined in detail:

1) **Vul(Step1):**
   - Communication System: GSM
   - Security Measure: Encryption and digital signature
   - Acquired Privilege: Read/Write
   - Violated Security Property: Confidentiality and authentication
   - Exploit: Bypassing encryption/signature algorithms and establishing a GSM network to access remote services offered by the Telematics unit.
2) **Vul(Step2a):**
   - Component: Telematics
   - Security Measure: None
   - Acquired Privilege: Full Control
   - Violated Security Property: Authorization

- Exploit: No access control implemented on the Telematics ECU, so attackers are authorized users while sending valid GSM messages.

3) **Vul(Step3a):**
   - Communication System: USB
   - Security Measure: None
   - Acquired Privilege: Read/Write
   - Violated Security Property: Authentication
   - Exploit: Sending valid messages from Telematics the USB channel.
4) **Vul(Step4a):**
   - Component: Infotainment
   - Security Measure: Digital signature of firmware
   - Acquired Privilege: Full Control
   - Violated Security Property: Integrity
   - Exploit: Exploiting a memory vulnerability in the Infotainment's in-vehicle browser.

For the given E/E architecture, our security model was able to describe this attack path, which was exploited by the researchers, on an abstract level. Thus, in general our approach is able to model realistic attack paths. If this attack path had already been identified during the development of the attacked vehicle, further measures could have been taken to secure the system. However, in order to use this approach to generate attack paths for security testing, a crucial aspect is the consideration of exploits the attackers apply during an attack as well as the vulnerabilities which are exploited. We have defined vulnerabilities for our model on a rather abstract level by the violated security property as well as the achieved privilege. This abstract view serves to identify all possible attack paths in the system. Once these paths are known, possible exploits can be attached to a vulnerability. We plan to use Microsoft's STRIDE classification [29] for this purpose. This approach assigns an attack technique to the six security properties we use to model our vulnerabilities. For example, the STRIDE category spoofing can be assigned to the security property authentication. The STRIDE categories can also be further broken down into more concrete attack techniques [1]. This incremental approach allows the attack paths to be successively refined as the components of the vehicle are specified more concretely during development. In this way, we can use our approach on different stages of the vehicle development process with different levels of abstraction.

V. Discussion

In this section, general aspects of our approach will be discussed in order to address current challenges and to show possible solutions. In Section III, we have shown how our security model can be generated from the E/E architecture of a vehicle, its implemented security measures, and attack characteristics, such as our privilege model. This model is intended to derive attack paths that can be used for security testing. In general, the identification of attack paths is very similar to performing a Threat Analysis and Risk Assessment (TARA). However, a TARA does not usually consider security requirements or security measures, as these are derived based on the results of a TARA. Our approach starts at this point and thus considers security measures when creating our security model. However, there is potential at this point to couple the process of TARA with our model-based security testing approach. In [27], we have already shown how our *Attacker Privilege* model can be used to automatically derive attack trees based on a formal model. Another aspect that should be considered in this context is the complexity of our security model. Since the architectures of modern vehicles can contain a large number of components, there is a risk that the security model could become very large and result in a large number of attack paths. At this point, a prioritization would have to take place in order to sort out attack paths with low priority. However, this also applies to other development activities, such as TARA, due to the high complexity of modern vehicles. Therefore, we could reuse risk values determined during TARA to prioritize attack paths. Alternatively, to reduce the complexity of our model, we can assume that the attacker has already achieved a certain privilege within the model. In this way, we would be able to generate attack paths only from certain parts of our model. Another challenge is the formalization of our approach. So far, this paper has outlined the basic process of creating security models and their graphical representation. We are currently working on formalizing our model as an EFSM. By formally describing an E/E architecture and attack characteristics, the modeling process can be automated and continuously refined during vehicle development. For example, in early development phases, an analysis of our model could be carried out using a vulnerability database. Colleagues of Karlsruhe University [30], [31] are currently working on the development of an automotive vulnerability database which could be used for that purpose. On further development stages the model can be refined and attack paths can be made more concrete to serve as actual security test cases. Since the focus of this paper was on the concept for generating the security model, potential methods for deriving the attack paths have not been discussed in detail so far. Depending on the size and complexity of the model, a path-based coverage criterion could therefore be used to derive attack paths. As an alternative, a derivation of attack paths could be done, for example, by model checking [32], where generated counterexamples are used as attack paths. We used this technique in [27] to generate attack trees. Another possibility could be SAT solving (Boolean Satisfiability Problem) [32].

VI. Conclusion and Future Work

In this paper, a model-based security testing approach for the automotive domain was presented. This approach enables automation in the security testing process at an early stage of vehicle development. Thus, security concepts and measures can be validated and potential vulnerabilities can be identified early in an automated way. On the one hand, this enables compliance with standards, such as ISO 21434 or UN R155. On the other hand, an early identification of vulnerabilities helps to secure the vehicle before it is on the road, which can increase the security and safety of vehicles in road traffic. This is particularly relevant for autonomous vehicles, as the driver no longer has the option of intervening if the vehicle is affected by a cyber attack. In order to reach that goal, we create a security model based on the E/E architecture of a vehicle by considering security-related external entities (including an attacker) as well as characteristics of security attacks. Since a security model can already be created in the design and concept phase, analyses of vulnerabilities are possible early on. We applied that process to a real-world vehicle architecture. The potential of our approach has been illustrated, since the resulting security model was able to identify an attack path which was used by a successful attack on that vehicle. For future work, we plan to formalize the security model. This enables a formal description of model elements in order to automatically create and analyze the security model. Another future challenge lies in the technique to derive attack vectors in an automated way (e.g., by model checking). Furthermore, a prioritization and refinement of attack vectors and their attack steps is a challenge for future work. Finally, in order to further evaluate our approach, a case study could be performed. Our collection of automotive security attacks [22] could serve as a database for this purpose.

REFERENCES

[1] F. Sommer, J. Dürrwang, and R. Kriesten, "Survey and classification of automotive security attacks," *Information*, vol. 10, no. 4, p. 148, 2019.

[2] ISO/SAE DIS 21434, "Road vehicles — cybersecurity engineering: Status : Under development," 2021.

[3] UNECE, "Un regulation no. 155 - uniform provisions concerning the approval of vehicles with regards to cyber security and cyber security management system: E/ece/trans/505/rev.3/add. 154," 03/2021. [Online]. Available: https://unece.org/sites/default/files/2021-03/R155e.pdf

[4] S. Marksteiner and Z. Ma, "Approaching the automation of cyber security testing of connected vehicles," in *Proceedings of the Third Central European Cybersecurity Conference*, 2019, pp. 1–3.

[5] A. Kasoju, K. Petersen, and M. V. Mäntylä, "Analyzing an automotive testing process with evidence-based software engineering," *Information and Software Technology*, vol. 55, no. 7, pp. 1237–1259, 2013.

[6] M. A. Khan, A. Jadoon, K. M. S. Haq, S. Mumtaz, and J. Rodrigues, "An overview of resilient and automatic model-based testing approaches for automotive industry," in *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2019, pp. 1–6.

[7] P. A. P. Salas, P. Krishnan, and K. J. Ross, "Model-based security vulnerability testing," in *2007 Australian Software Engineering Conference (ASWEC'07)*, 2007, pp. 284–296.

[8] A. Andrews, M. Abdelgawad, and A. Gario, "World model for testing autonomous systems using petri nets," in *The 17th IEEE International Symposium on High Assurance Systems Engineering*, R. Babiceanu, H. Waeselynck, R. A. Paul, B. Cukic, J. Xu, and I. I. S. H. A. S. Engineering, Eds. Piscataway, NJ: IEEE, 2016, pp. 65–69.

[9] S. Kriebel, M. Markthaler, K. S. Salman, T. Greifenberg, S. Hillemacher, B. Rumpe, C. Schulze, A. Wortmann, P. Orth, and J. Richenhagen, "Improving model-based testing in automotive software engineering," in *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*, 2018, pp. 172–180.

[10] I. Pekaric, C. Sauerwein, and M. Felderer, "Applying security testing techniques to automotive engineering," in *Proceedings of the 14th International Conference on Availability, Reliability and Security*, 2019, p. 61.

[11] M. Felderer, B. Agreiter, P. Zech, and R. Breu, "A classification for model-based security testing," *Advances in System Testing and Validation Lifecycle (VALID 2011)*, pp. 109–114, 2011.

[12] M. Büchler, "Semi-automatic security testing of web applications with fault models and properties," Phd Thesis, Technische Universität München, München, November 3, 2020.

[13] E. Martin, "Automated test generation for access control policies," in *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*, 2006, pp. 752–753.

[14] S. Hagerman, A. Andrews, and S. Oakes, "Security testing of an unmanned aerial vehicle (uav)," in *2016 Cybersecurity Symposium (CYBERSEC)*, 2016, pp. 26–31.

[15] A. D. Brucker, L. Brügger, P. Kearney, and B. Wolff, "An approach to modular and testable security models of real-world health-care applications," in *Proceedings of the 16th ACM symposium on Access control models and technologies*, 2011, pp. 133–142.

[16] E. LeMay, W. Unkenholz, D. Parks, C. Muehrcke, K. Keefe, and W. H. Sanders, "Adversary-driven state-based system security evaluation," in *Proceedings of the 6th International Workshop on Security Measurements and Metrics*, 2010, pp. 1–9.

[17] M. Cheah, H. N. Nguyen, J. Bryans, and S. A. Shaikh, "Formalising systematic security evaluations using attack trees for automotive applications," in *IFIP International Conference on Information Security Theory and Practice*, 2017, pp. 113–129.

[18] ISO 11898-1:2015, "Road vehicles – controller area network (can) – part 1: Data link layer and physical signalling," 1993.

[19] P. Mundhenk, S. Steinhorst, M. Lukasiewycz, S. A. Fahmy, and S. Chakraborty, "Security analysis of automotive architectures using probabilistic model checking," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2015, pp. 1–6.

[20] P. S. Oruganti, M. Appel, and Q. Ahmed, "Hardware-in-loop based automotive embedded systems cybersecurity evaluation testbed," in *Proceedings of the ACM Workshop on Automotive Cybersecurity*, 2019, pp. 41–44.

[21] T. Volkersdorfer and H.-J. Hof, "A concept of an attack model for a model-based security testing framework: Introducing a holistic perspective of cyberattacks in software engineering," *SECURWARE 2020 : The Fourteenth International Conference on Emerging Security Information, Systems and Technologies*, 2020.

[22] F. Sommer and J. Dürrwang, "Ieem-hska/aad: Automotive attack database (aad)," 2019. [Online]. Available: https://github.com/IEEM-HsKA/AAD

[23] Bluetooth Special Interest Group, "Bluetooth core specification v5.0," https://www.bluetooth.com/specifications/bluetooth-core-specification, 2016. [Online]. Available: https://www.bluetooth.com/specifications/bluetooth-core-specification

[24] ISO 26262-1:2018, "Road vehicles – functional safety: Part 1: Vocabulary," Dezember 2018.

[25] AUTOSAR, "Specification of secure onboard communication," 2018. [Online]. Available: https://www.autosar.org/fileadmin/user_upload/standards/classic/4-3/AUTOSAR_SWS_SecureOnboardCommunication.pdf

[26] M. Rumez, A. Duda, P. Gründer, R. Kriesten, and E. Sax, "Integration of attribute-based access control into automotive architectures," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 1916–1922.

[27] J. Dürrwang, F. Sommer, and R. Kriesten, "Automation in automotive security by using attacker privileges," *Proceedings 19th escar Europe 2021*, 2021.

[28] Keen Security Lab, "Experimental security assessment of bmw cars: A summary report," 2017. [Online]. Available: https://keenlab.tencent.com/en/whitepapers/Experimental_Security_Assessment_of_BMW_Cars_by_KeenLab.pdf

[29] L. Kohnfelder and P. Garg, "The stride threat model," 2009. [Online]. Available: https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20)

[30] R. Bolz, M. Rumez, F. Sommer, J. Dürrwang, and R. Kriesten, "Enhancement of cyber security for cyber physical systems in the automotive field through attack analysis," *embedded world Conference 2020 Proceedings*, 2020. [Online]. Available: https://www.researchgate.net/publication/339643941_Enhancement_of_Cyber_Security_for_Cyber_Physical_Systems_in_the_Automotive_Field_Through_Attack_Analysis

[31] R. Bolz and R. Kriesten, "Automotive vulnerability disclosure: Stakeholders, opportunities, challenges," *Journal of Cybersecurity and Privacy*, vol. 1, no. 2, pp. 274–288, 2021.

[32] E. M. Clarke, T. A. Henzinger, H. Veith, and R. Bloem, *Handbook of model checking*. Springer, 2018.