Automation in Automotive Security by Using Attacker Privileges

Jürgen Dürrwang[0000−0001−6045−1565], Florian Sommer[0000−0002−4009−7164], and Reiner Kriesten

Institute of Energy Efficient Mobility (IEEM),
Karlsruhe University of Applied Sciences,
Moltkestr. 30, 76133 Karlsruhe, Germany
juergen.duerrwang@outlook.de,
{florian.sommer, reiner.kriesten}@h-ka.de
https://www.h-ka.de/en/ieem

Abstract. Modern vehicles contain a large number of electronic information technology components which are increasingly connected to the outside world. This results in a higher risk for possible cyber attacks. To prevent such attacks, threat and risk analyses and comprehensive security tests are carried out during the development of a vehicle in order to identify and mitigate potential vulnerabilities. However, these processes are usually carried out manually. Due to the increasing complexity of modern vehicles, manual analysis and test methods reach their limits. For this reason, we present an approach of modeling attacker privileges which are used to automate the threat and risk analysis as well as the security testing process. Thereby, we illustrate how these privileges are applied to formalize a vehicle's internal network. We use this formal model to generate attack trees and security test cases. Furthermore, we show the application of our approach on an exemplary vehicle network and illustrate how to derive attack trees by model checking techniques in an automated way.

Keywords: Attacker Privileges · Threat and Risk Analysis · Security Testing · Automation

1 Introduction

Modern vehicles consist of a high number of information technology components which operate in an increasingly connected and complex manner. This complexity reaches its peak in autonomous driving. Hence, a safe operation of vehicles has to be ensured in addition to their reliable functionality. This concerns both the vehicle's functional safety as well as its cyber security. The feasibility of security attacks has been demonstrated in recent years by various researchers [28]. Since a cyber attack can endanger the health of vehicle occupants and surrounding traffic, protection against such attacks has increasingly become the focus of automotive manufacturers and suppliers. To integrate security activities into the vehicle development process, standards, such as the upcoming ISO 21434 [14], and regulations, such as UN R155 [30], have to be considered by Original Equipment Manufacturers (OEMs) and suppliers. These standards especially require the execution of a Threat Analysis and Risk Assessment (TARA) [24] as well as a comprehensive security testing of a vehicle and its components in order to identify and mitigate potential vulnerabilities. While security experts analyze vehicles and its subsystems for possible threats and their risks during the TARA, security testing (e.g., penetration testing) is carried out to uncover vulnerabilities by executing attacks.

Problem: Since both processes depend on the experience of the security experts and testers, they are often carried out manually. Thus, these activities are time-consuming and generally repeated for every new component in a vehicle. Due to the increasing complexity of modern vehicles, the time and resources required to perform these activities are also increasing, which is a major challenge in automotive threat analysis [25] and testing [15]. As a result, manual methods are reaching their limits in modern vehicles [20].

Solution: In this paper, we present an approach to automate the processes of TARA and security testing. For this purpose, we present the concept of *Attacker Privileges* to describe states in which an attacker can perform certain attacks in a vehicle network. Based on these privileges, we create a formal vehicle network model which is used to automatically generate attack trees in a TARA as well as attack paths for security testing.

Contribution: We illustrate how we apply these privileges to automotive-related security attacks, which were carried out in the past. Furthermore, this paper shows an approach to formally describe a vehicle's internal network as a transition system based on our privileges. We integrated a model checker in a software tool which uses this formal model to automatically generate attack trees. An exemplary application of our approach is presented resulting in an attack tree including attack paths which has been exploited in a real airbag system [7]. Finally, we illustrate how the privilege model is used for security testing by deriving attack paths.

This paper is structured as follows: In Section 2, we describe the fundamentals of security testing and TARA. Section 3 introduces the concept of *Attacker Privileges* and shows application examples on real-world automotive security attacks. In Section 4, we show how these privileges are used to automate the process of TARA and security testing. In Section 5, a conclusion is drawn and an outlook on future work is given.

2 Background and Related Work

In this section, a brief introduction to TARA and security testing is provided.

2.1 Threat Analysis and Risk Assessment (TARA)

The goal of a TARA [24] is to identify potential security threats to a system and assess their risk. Based on these results, concepts and countermeasures are derived in order to mitigate those threats. A TARA consists of the following steps. First, components and system areas are specified, which should be addressed by this analysis. In the automotive domain, for example, this can be an Electronic Control Unit (ECU), which has an impact on safety, such as airbags or steering ECUs. Furthermore, assets, such as safety or financial assets, are identified which have to be protected against threats. Based on this system scope, possible threats are identified, which could occur in the system. In the automotive domain, various methods have been established for this purpose, such as E-safety Vehicle Intrusion Protected Applications (EVITA) [9] or HEAling Vulnerabilities to ENhance Software Security and Safety (HEAVENS) [18]. Threats are often represented as attack trees [26] in which possible threats to reach a certain attack goal are described in a tree structure. The final step of a TARA consists of a risk assessment. Risk is usually defined by the probability of occurrence of a threat as well as its impact on a system. For example, ISO 21434 suggests Common Vulnerability Scoring System (CVSS) [6] as a possible metric for risk assessment. In comparison to these approaches, our method is able to automate a TARA by generating attack trees automatically based on the privilege model introduced in this paper.

2.2 Security Testing

Security testing is carried out to ensure the correctness of integrated security measures, such as encryption, and to identify potential remaining vulnerabilities in a system [8]. In the automotive domain, ISO 21434 [14] suggests penetration testing [1] and dynamic analyses, such as fuzzing and vulnerability scanning, for this purpose. However, these test techniques usually require fully developed systems for test execution. Thus, they can only be applied in late stages of development. Furthermore, test methods, such as penetration testing, are usually carried out manually, since they represent exploratory methods. Thus, approaches have been published which enable automation in security testing. For example, Cheah et al. [4] present an approach for an automatic generation of security test cases from attack trees by using model checking. Their approach is mainly applied on automotive communication systems, such as Bluetooth [3]. Other approaches such as [22] focus on automated security testing in virtual environments. Instead of individual components and communication systems, we take the whole vehicle network as well as its interfaces including communication with external entities into consideration. Thus, the privilege model presented in this paper is used for security testing of attack paths through the vehicle network.

3 Attacker Privileges

To enable the execution of attacks or attack steps in vehicles, an attacker must have certain privileges, which make these attacks possible. For example, an attacker must have access to a component or interface in order to execute certain functions or exploit vulnerabilities. To describe such privilege levels, we introduced a privilege model in [28], which consists of five elements:

- Read/Write (Functional Communication Link)
- Execute (Functional Component)
- Read (Functional Component)
- Write (Functional Component)
- Full Control (Functional Component)

These privileges do not correspond to classical privilege management or access control systems. They are rather used to describe a certain state in a system under attack which characterizes actions an attacker can perform after acquiring such a privilege. For this reason, we define them as *Attacker Privileges*. In this paper, we go into more detail about these privileges and show how we use them in order to create models which are used to automate TARA and security testing activities.

3.1 Read/Write (Functional Communication Link)

This privilege only applies to communication systems. In general, communication systems are accessible for reading and writing as soon as there is a physical access to it, especially when there are no security mechanisms implemented. For example, if an attacker has access to the Controller Area Network (CAN) [12] interface of a vehicle, reading and writing messages on this channel will be possible. In this case, Read/Write (Functional Communication Link) is acquired. However, if there are security measures implemented, such as encryption, an attacker can also read and write messages, but will not be able to understand its content, so the Read/Write privilege is not acquired. Another possible security measure in the automotive domain is SecOC [2] which is used to authenticate communication partners. If an attacker is not able to create a SecOC-compliant authentication code, the Read/Write privilege will not be acquired. We only assign this privilege when an attacker is able to communicate with a vehicle and is able to interpret message contents. We also include physical signals (for example, in physical attacks) in the class of communication channels, which can take that privilege. Furthermore, we do not distinguish between read and write at this point. Although it would theoretically be possible to read without writing, an attacker would not get further within the vehicle network without the ability to write on this communication channel. However, a more precise distinction between read and write could be made in security testing, since more details are required here than in a TARA. In Figure 1, an example of acquiring the Read/Write (Functional Communication Link) privilege is illustrated. In this example, an attacker has access to the On-Board Diagnostics (OBD) interface of the vehicle, which is connected to a CAN-Bus. Since there is no security measure implemented for CAN, the attacker is able to monitor traffic as well as sending malicious messages. A similar attack was carried out by Miller and Valasek [21] in order to manipulate the instrument clusters of two vehicles. As a result, it was possible to display false speeds, change the mileage counter, and activate various warning lights. Similar attacks are described in [10, 28].

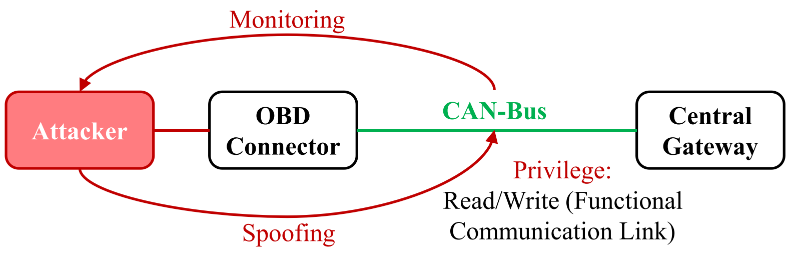

Fig. 1: Read/Write (Functional Communication Link) privilege acquired by monitoring and spoofing CAN-Bus messages.

3.2 Execute (Functional Component)

For vehicle components (e.g., ECUs), we assign four different privilege levels (Execute, Read, Write, Full Control). The Execute (Functional Component) privilege is achieved when an attacker is able to trigger functions of a component. In this case, an attacker does not need any information about a component and its behavior, but is able to execute functions anyway by sending messages or other inputs. For example, an attacker sends diagnostic messages to an ECU in order to trigger diagnostic services, such as controlling actuators or resetting the ECU, as it is shown in Figure 2. An attacker is connected to the OBD Connector in order to trigger diagnostic functions of the Central Gateway. A high number of attacks, which were carried out in the past [27], were targeting the execution of diagnostic functions. As a result, attackers were able to shut down a vehicle's engine [21], deactivate ECU communication [17], and compromise ECUs [16].

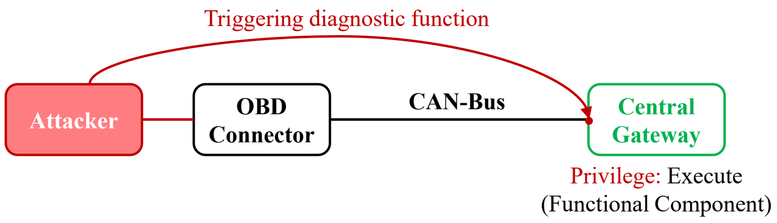
Triggering diagnostic function
Attacker
OBD Connector
CAN-Bus
Central Gateway
Privilege: Execute (Functional Component)

Fig. 2: Execute (Functional Component) privilege acquired by unauthorized triggering of diagnostic messages.

If there are any security measures, such as *Security Access* [13], implemented on a component, which prevent that execution, an attacker can't acquire the Execute privilege. If an attacker is able to overcome this measure, the Execute privilege will be achieved. It is also possible that both secured and unsecured functions are implemented on a component. Since we use our privileges to find attack paths through the vehicle network, we do not initially distinguish between these functions. However, this distinction can be made in the further course (for example, during testing).

3.3 Read (Functional Component)

The Read (Functional Component) privilege is used to describe that an attacker is able to read information or data from a component. For example, the Read privilege will be acquired, if an ECU's firmware is downloaded or sensitive data of a driver is read. In Figure 3, an example of acquiring the Read (Functional Component) privilege is shown.

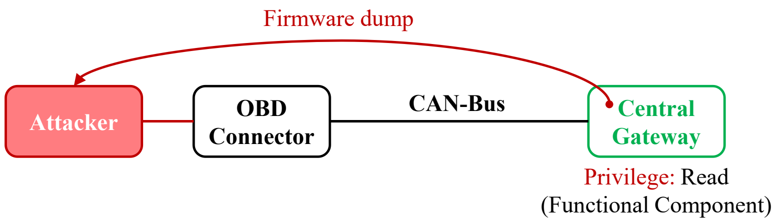

Fig. 3: Read (Functional Component) privilege acquired by extracting the firmware of an ECU.

In this example, an attacker is connected to the OBD interface and extracts the firmware from a Central Gateway. The Read (Functional Component) privilege was acquired by a number of real-world attacks in the past. In a lot of cases attackers were able to extract an ECU's firmware [5, 16, 21]. Other attacks lead to a discovery of an ECU's memory content [17] or security access keys [17, 21].

3.4 Write (Functional Component)

The Write (Functional Component) privilege will be acquired if an attacker is able to modify data of a component, for example, modifying specific parts of an ECU's firmware or conducting a buffer overflow to change application data. In Figure 4, an example of acquiring the Write (Functional Component) privilege is shown. In this example, an attacker is again connected to the OBD Connector of a vehicle and able to change the vehicle's identification number on the Central Gateway. Usually, such attacks require overcoming the Security Access [13] of an ECU which is a diagnostic security measure to prevent unauthorized actions on a component.

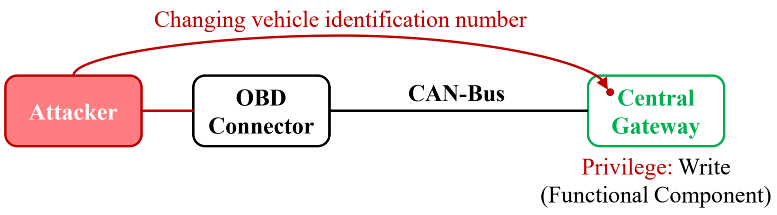
Changing vehicle identification number
Attacker
OBD Connector
CAN-Bus
Central Gateway
Privilege: Write (Functional Component)

Fig. 4: Write (Functional Component) privilege acquired by changing a vehicle's identification number.

Examples of real-world attacks, which led to the Write (Functional Component) privilege, consist of buffer overflows [5] or modification of firmware [21].

3.5 Full Control (Functional Component)

The Full Control (Functional Component) privilege will be acquired if an attacker is able to completely take over a vehicle component. This is similar to root privileges in Information Technology (IT). A possible scenario is flashing a malicious firmware update on an ECU. If that privilege is acquired, the attacker can basically do everything with an ECU including sending malicious messages to other ECUs, or triggering actuators, such as airbags or steering. In Figure 5, an example of acquiring the Full Control (Functional Component) privilege is illustrated. In this example, an attacker is able to flash a modified, malicious firmware update on the Central Gateway and obtains the Full Control (Functional Component) privilege. The feasibility of flashing malicious firmware updates was shown in several real-world attacks [17,21]. Other attacks consisted of exploiting vulnerabilities in infotainment ECUs [16, 19] to acquire this privilege.

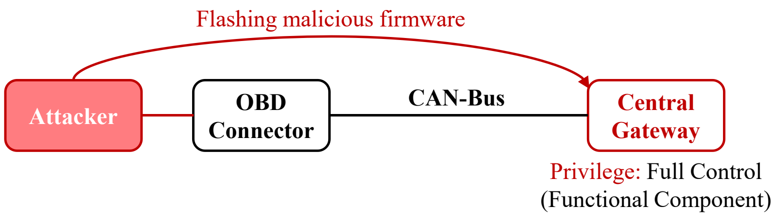
Flashing malicious firmware
Attacker
OBD Connector
CAN-Bus
Central Gateway
Privilege: Full Control (Functional Component)

Fig. 5: Full Control (Functional Component) privilege by flashing a malicious firmware on the Central Gateway.

3.6 Proof of Concept and Analysis

In order to evaluate the *Attacker Privileges*, we applied them to our database of automotive security attacks [27]. This collection consists of 162 attacks conducted and published between 2002 and 2019. These attacks were divided into their individual attack steps (412 steps in total) and classified by a security attack taxonomy [28]. The application of our privilege model to that database provided the following results: Most attack steps resulted in an obtainment of the Execute privilege (177 attack steps). This is due to the fact that many attacks aimed to trigger certain functions such as diagnostic services by sending messages to an ECU. This class also includes replay attacks. Further, a high number of steps led to the Read/Write privilege (96 attack steps) and can be explained by the fact that the initial action of an attack is to establish a connection to the vehicle, for example, via CAN. In most cases, the Read privilege (86 attack steps) was achieved by extracting the firmware of an ECU or by reading ECU data. The Write privilege (15 attack steps) was mainly acquired by manipulating data on components, whereas the Full Control privilege (38 attack steps) was obtained by flashing complete malicious firmware updates to ECUs. Overall, we were able to assign the privilege model to all of the 412 attack steps from the attack database. These attacks cover a wide range of scenarios (local and remote attacks), which includes most existing automotive components and communication systems as well as a high number of different attack techniques, Thus, it can be concluded that our *Attacker Privileges* are appropriate to describe a wide range of security attacks on automotive systems. However, there is still a risk that the privileges are not sufficient to describe certain attacks in all its details. For this reason, we take additional attack-relevant parameters into account to create models for TARA and security testing. This process is introduced in the next section.

4 Attacker Privileges in TARA and Security Testing

This section shows the application of *Attacker Privileges* within a TARA for automated attack tree generation. Furthermore, it is shown how privileges are applied for security testing.

4.1 Attacker Privileges in Vehicle Networks

Attacker Privileges describe abstract states of a system in which an attacker can perform certain actions. We use this circumstance in a TARA and in security testing to model which attack steps are necessary to reach a certain goal, such as compromising an ECU. A similar approach was presented by Hoppe et al. [11] by introducing five attack principles (read, modify, interrupt, create/spoof, steal/remove), which are used to model threats. This approach takes a black box view and shows an application of privacy attacks for threat analyses and an exemplary test environment. In comparison to this work, our approach is white box method, since it is applied in the concept and design phase of the vehicle development process. However, since the level of detail at this stage is rather low, our privileges are used to describe general states an attacker can reach in a vehicle network instead of actual attack techniques. The use of our privileges to model attack paths through vehicle networks in particular to automate TARA and security testing can be considered a novelty. To accomplish this goal, we interpret an attack as a path or sequence of attack steps, whereby each successfully executed attack step (exploiting a vulnerability) leads to an *Attacker Privilege*. In order to illustrate this process, Figure 6 presents a scenario in which two communication systems are connected to a component.

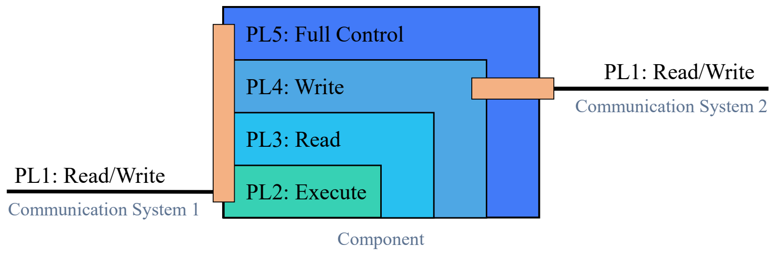

Fig. 6: Privilege Levels (PL) and their mapping to a component or communication system.

Assuming that an attacker has the *Read/Write* privilege (PL 1) on Communication System 1, communication with the connected component is possible. Thus, an attacker can achieve one of the component privileges (PL 2, 3, 4, or 5) by exploiting another vulnerability. It is possible for an attacker to reach PL 4 directly from PL 1, for example, by modifying data on an ECU through a buffer overflow attack. When PL 4 or PL 5 is reached, it is in principle possible for the attacker to access a component's communication interfaces and reach PL 1 on further connected communication systems. Furthermore, it is possible to switch between privileges arbitrarily, i.e. an attacker can also try to get from PL4 to PL2. Thus, there is not necessarily a hierarchy between the privileges, although it can be argued that the *Full Control* privilege includes the other privileges. However, a lot of attacks in our collection [27] followed that hierarchy in which PL 2 is the lowest and PL 5 is the highest privilege. For this reason, we adopted that order in particular for the TARA approach. As a result, for this work we only consider attacks which lead to higher privileges. Nevertheless, it is also possible to create the TARA model without using that hierarchy. The method introduced in this section is used to describe attack paths within a vehicle network. For example, an attacker could get access to an infotainment ECU by its Bluetooth interface. From there, the attacker could go further to access the central gateway via Ethernet through various attack steps in order to exploit other components, such as an airbag ECU.

4.2 Attacker Privileges in TARA

In this section, we show how *Attacker Privileges* are used to automatically create attack trees during a TARA. For this purpose, the item under consideration (e.g., ECU), vehicle network (E/E architecture), and vulnerabilities from our automotive security attack database [27,28] are transferred into a formal model. To be precise, a transition system is created whose state space corresponds to an exploitation of the vulnerabilities in the respective components. To build that model, vulnerabilities from the database are assigned to components and communication links of the E/E architecture. This also enables an assignment of vulnerabilities to new systems which have not been analyzed before. In Figure 7, the principle approach for our formal model is illustrated.

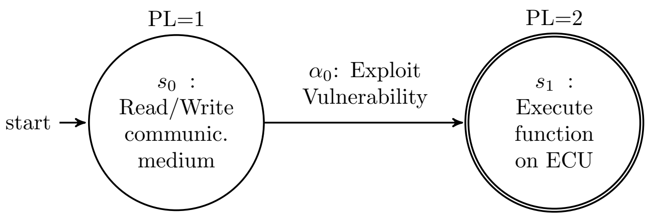
PL=1
PL=2
start
s_0 : Read/Write communic. medium
α_0: Exploit Vulnerability
s_1 : Execute function on ECU

Fig. 7: Simple transition system with two states (s_0, s_1) and one transition (α_0) describing the exploitation of a vulnerability on an ECU.

The transition α_0 represents an exploitation of a vulnerability. The combination of the two states (s_0, s_1) and the transition describe a successfully executed attack step. An attacker has to fulfill certain preconditions in order to be able to exploit a vulnerability. For this purpose, we consider our privilege levels (Section 3) as preconditions and specify their fulfillment implicitly in the transitions. Thus, an attacker has to achieve a certain privilege level (state) to be able to exploit a vulnerability (transition). Once it is exploited, an attacker can achieve a new and higher level which enables him to exploit further vulnerabilities in a system. Achieved Privelege Levels (PLs) are shown as state labels with PL=1 for s_0 and PL=2 for s_1. In state s_0, the attacker owns PL 1 and, after exploitation, he achieves PL 2 which enables an execution of functions, e.g., a diagnostic function. Our proposed model is created from an attacker's point of view and describes possible privilege levels which are required to exploit a vulnerability and subsequently to obtain a higher privilege level. With this approach, it is possible to model a chain of privilege escalations across multiple elements in a vehicle network. In combination with the transition system concept in Figure 7 and the privilege model, attack paths or attack graphs can automatically be generated. For this purpose, we apply model-checking techniques [23]. A model checker verifies if an analyzed model violates a certain specification. If a specification is violated, the model checker will produce a counterexample. For the E/E architecture in Figure 8, a specification could be: *The integrity of an airbag* *control unit should never be violated.* A violation of integrity corresponds to a counterexample the model checker produces. This counterexample characterizes a path in the transition system, which leads from a starting point to a violation of integrity on the target. Thus, the counterexample represents a potential attack path in the vehicle network. In the following, we apply our approach to a real-world example.

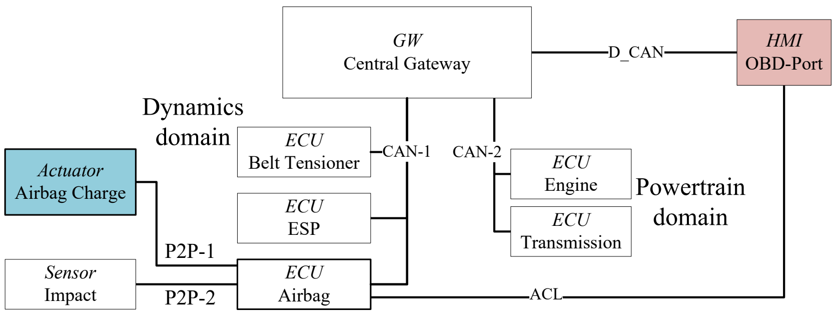

Fig. 8: Examplary E/E architecture, which was used for evaluating the attack tree generation. The OBD-Port is used as a starting point, whereas Airbag Charge is the attack target. Abbreviations: D_CAN (Diagnostic CAN), ACL (Additional Communication Line), P2P (Peer-to-Peer).

Therefore, we take the E/E architecture of Figure 8 in consideration and apply vulnerabilities from our attack database [27] to the components and communication links. An extract of these vulnerabilities is given in Table 1 and correspond to vulnerabilities identified in a real attack on an airbag control unit [7]. We use the OBD-Port (which is interpreted as a component) as the attacker's starting point and the pyrotechnic Airbag Charge of the Airbag ECU as the attack goal. Based on the E/E architecture a formal transition system is created according to Figure 7. In order to automate the creation of the model, the Automotive-Security-Threat-Modeling-Tool (ASTMT) was developed. This software tool takes the E/E architecture, attack starting point and attack target as well as the corresponding vulnerabilities as inputs. A transition system is built automatically which uses the privilege model from Figure 6 and corresponds to the logic of Figure 7. For an automatic generation of an attack tree from that transition system, we included a model checker [29] which is capable of identifying every possible counterexamples (paths) in a specification. The transition system and specification are passed to the model checker. For this example, it was specified that the authenticity of the Airbag's Pyrotechnic Charge should never be violated, since only authorized people should have access to an airbag charge in order to detonate it. However, the automatically generated attack tree in Figure 9 shows that there are two paths which allow unauthorized firing of airbag charges.

Table 1: Extract of vulnerabilities used for the evaluation (based on our data collection [28]) with their respective privilege levels and CVSS exploitability.

Description	Required privilege level	Acquired privileges level	Element	Exploitability CVSS
Brute forcing Security Access	1	2	Airbag-ECU	2,84
Exploiting vulnerability to deploy Airbag	2	4	Airbag-ECU	2,07
Missing authorization for using OBD	1	5	OBD	0,92
General relay of diagnostic data on P2P	1	2	P2P	0,92
Relay of diagnostic messages by gateway	2	4	Gateway	2,84
General relay of diagnostic data on CAN	1	2	CAN	2,84
Missing authorization for using actuator	2	2	Actuator	2,52

Starting at the OBD-Port, exploited vulnerabilities from Table 1 are mapped to edges of the paths, whereas nodes represent elements from the E/E architecture. The right path corresponds to an attack path followed by attackers in reality [7]. The left path was confirmed as a feasible attack in a later analysis, meaning that the tool was able to discover not-yet-known attacks. Additionally, the tool is also able to calculate the likelihood of occurrence of paths.

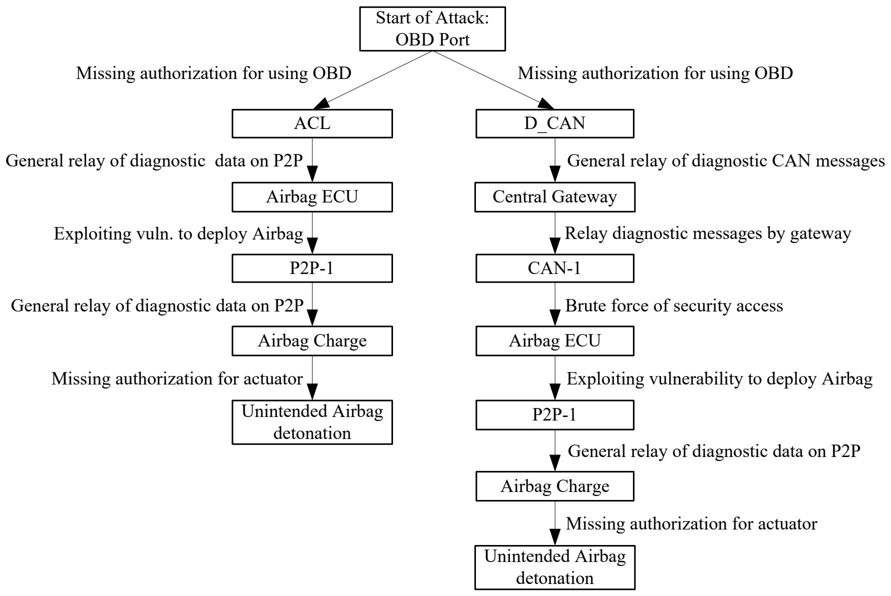
Start of Attack: OBD Port
Missing authorization for using OBD
Missing authorization for using OBD
ACL
D_CAN
General relay of diagnostic data on P2P
General relay of diagnostic CAN messages
Airbag ECU
Central Gateway
Exploiting vuln. to deploy Airbag
Relay diagnostic messages by gateway
P2P-1
CAN-1
General relay of diagnostic data on P2P
Brute force of security access
Airbag Charge
Airbag ECU
Missing authorization for actuator
Exploiting vulnerability to deploy Airbag
Unintended Airbag detonation
P2P-1
General relay of diagnostic data on P2P
Airbag Charge
Missing authorization for actuator
Unintended Airbag detonation

Fig. 9: Example for an attack tree by using *Attacker Privileges* to model the impact an attack has on a component or communication system.

For this purpose, a partial metric of CVSS [6] is used, which describes the exploitability of each vulnerability and is provided by the data collection (Table 1). In this way, individual attack paths can be prioritized by their exploitability in order to identify critical paths which have to be protected by security measures, such as access controls or Secure Onboard Communication (SecOC) [2].

4.3 Attacker Privileges in Security Testing

In this section, we show how *Attacker Privileges* are used to automate the test process by creating a model of a vehicle's E/E architecture in order to automatically generate attack paths from that model. Theses attack vectors are used as test paths during security testing. When carrying out security tests on a vehicle, testers usually act like attackers, since both parties aim to find vulnerabilities. The tester tries to access a vehicle and its components via an interface and subsequently compromise components or get to a target component through the network. This enables the use of our privileges model described in Section 4.1 to model attacks or attack paths through the vehicle's E/E architecture. In Figure 10, the subset of an attack path through the vehicle network is shown as an example. The example shows a tester having access to the CAN-Bus via the OBD interface of a vehicle and being able to access the Central Gateway. For each access, a tester must perform an attack step to exploit a vulnerability, which leads to a specific consequence. This consequence consists of an *Attacker Privilege*, which is achieved by the attack step and violated security property.

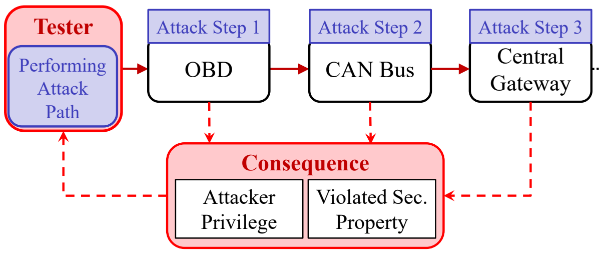

Fig. 10: Example for a test path by using *Attacker Privileges* to model which effect an attack has on a component or communication system.

In order to describe an attack step, we take following elements into account:

1. **Component** or **Communication System** which is attacked.
2. **Security Measure:** If a component or a communication system of the vehicle network is secured via a security measure, this will be taken into account, since a new privilege can only be achieved if a corresponding vulnerability and exploit exists to bypass this measure.
3. **Vulnerability:** At each attack step, a vulnerability must be exploited by violating a security property to achieve another *Attacker Privilege*.

This modeling approach can be used to automatically generate attack paths through vehicle networks similar to the approach for TARA (Section 4.2). For this purpose, a vehicle's E/E architecture is used to create a formal system model (for example, as an automaton) which is extended by applying our *Attacker Privileges* to individual components and communication systems. Based on that model, attack paths are generated (for example, by using model checking). These paths can be used as test cases during security testing. In comparison to our approach for building a formal model during a TARA (Section 4.2), the security testing approach takes security measures into account. Since that approach is still at an early stage, we could only provide a general overview here. Even though both approaches are based on the *Attacker Privileges*, the TARA and testing approach are standalone methods. Thus, a challenge for future work is to combine both approaches, for example, by using paths of the attack tree with a high risk as an input for the security testing approach. In this way, critical paths from TARA could be refined for testing. Another limitation of the security testing approach is that the automation does not include an automated test execution. Thus, the tester has to validate whether vulnerabilities, which are included in the model, exist on a real system and whether the modeled *Attacker Privileges* are achieved. Currently, our approach addresses an automated generation of the model as well as an automated generation of attack paths. Simulations in a virtual environment could validate the feasibility of such paths. For this purpose, our attack collection could be used to apply attacks (or attack paths) on our model. In this way, existing knowledge of attacks and vulnerabilities can be used for new systems from which we do not know if there are any vulnerabilities or security-related problems. However, since our security testing approach is still at an early stage, these challenges have to be addressed in future work.

5 Conclusion and Future Work

Due to the high complexity and the large number of different components and technologies in modern vehicles, ensuring vehicle security is a major challenge for manufacturers and suppliers. Development activities such as a TARA and security testing activities such as penetration testing are reaching their limits, as they are usually manual processes which are time consuming and costly. To automate these processes, we introduced the approach of *Attacker Privileges* which describe certain states in a system. These states characterize which actions an attacker can perform during an attack or attack step. We applied our privilege model to real-world automotive security attacks in order to show its application in practice. We also applied this concept to the TARA process by creating a formal model of a vehicle network based on the *Attacker Privileges*. Furthermore, we illustrated the automatic generation of attack trees based on that formal model by using a model checker in a custom software tool. Finally, we presented an application of our privileges in security testing by describing attack paths (test paths). For future work, we plan to formalize the security testing approach to enable early testing during development. Additionally, we plan to evaluate the TARA and security testing approach in a case study.

Acknowledgement

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

References

1. Arkin, B., Stender, S., McGraw, G.: Software penetration testing. IEEE Security & Privacy **3**(1), 84–87 (2005)
2. AUTOSAR: Specification of secure onboard communication, https://www.autosar.org/fileadmin/user_upload/standards/classic/4-3/AUTOSAR_SWS_SecureOnboardCommunication.pdf
3. Bluetooth Special Interest Group: Bluetooth core specification v5.0 (2016), https://www.bluetooth.com/specifications/bluetooth-core-specification
4. Cheah, M., Nguyen, H.N., Bryans, J., Shaikh, S.A.: Formalising systematic security evaluations using attack trees for automotive applications. In: IFIP International Conference on Information Security Theory and Practice. pp. 113–129 (2017)
5. Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S., Koscher, K., Czeskis, A., Roesner, F., Kohno, T., et al.: Comprehensive experimental analyses of automotive attack surfaces. In: USENIX Security Symposium (2011)
6. CVSS Special Interest Group: Common vulnerability scoring system v3.0: Specification document (2019), https://www.first.org/cvss/specification-document
7. Dürrwang, J., Braun, J., Rumez, M., Kriesten, R., Pretschner, A.: Enhancement of automotive penetration testing with threat analyses results. SAE International Journal of Transportation Cybersecurity and Privacy **1**(11-01-02-0005), 91–112 (2018)
8. Felderer, M., Büchler, M., Johns, M., Brucker, A.D., Breu, R., Pretschner, A.: Security testing: A survey. In: Memon, A. (ed.) Advances in computers vol. 101, Advances in Computers, vol. 101, pp. 1–51. Elsevier, Amsterdam (2016). https://doi.org/10.1016/bs.adcom.2015.11.003
9. Henniger, O., Apvrille, L., Fuchs, A., Roudier, Y., Ruddle, A., Weyl, B.: Security requirements for automotive on-board networks. In: 2009 9th International Conference on Intelligent Transport Systems Telecommunications,(ITST). pp. 641–646 (2009)
10. Hoppe, T., Kiltz, S., Dittmann, J.: Security threats to automotive can networks – practical examples and selected short-term countermeasures. In: Harrison, M.D., Sujan, M.A. (eds.) Computer safety, reliability, and security, LNCS sublibrary. SL 2, Programming and software engineering, vol. 5219, pp. 235–248. Springer, Berlin and New York (2008). https://doi.org/10.1007/978-3-540-87698-4_21
11. Hoppe, T., Kiltz, S., Dittmann, J.: Automotive it-security as a challenge: Basic attacks from the black box perspective on the example of privacy threats. In: International Conference on Computer Safety, Reliability, and Security. pp. 145–158 (2009)

12. ISO 11898-1:2015: Road vehicles – controller area network (can) – part 1: Data link layer and physical signalling (1993)
13. ISO 14229:2006: Road vehicles — unified diagnostic services (uds) — specification and requirements
14. ISO/SAE DIS 21434: Road vehicles — cybersecurity engineering: Status : Under development (2021)
15. Kasoju, A., Petersen, K., Mäntylä, M.V.: Analyzing an automotive testing process with evidence-based software engineering. Information and Software Technology **55**(7), 1237–1259 (2013)
16. Keen Security Lab: Experimental security assessment of bmw cars: A summary report (2017), https://keenlab.tencent.com/en/whitepapers/Experimental_Security_Assessment_of_BMW_Cars_by_KeenLab.pdf
17. Koscher, K., Czeskis, A., Roesner, F., Patel, S., Kohno, T., Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S.: Experimental security analysis of a modern automobile. In: 2010 IEEE Symposium on Security and Privacy. pp. 447–462. IEEE (5/16/2010 - 5/19/2010). https://doi.org/10.1109/SP.2010.34
18. Lautenbach, A., Islam, M.: Heavens–healing vulnerabilities to enhance software security and safety: Deliverable d2 - security models. The HEAVENS Consortium (2016)
19. Mahaffey, K.: Hacking a tesla model s: What we found and what we learned (2015), https://blog.lookout.com/hacking-a-tesla
20. Marksteiner, S., Ma, Z.: Approaching the automation of cyber security testing of connected vehicles. In: Proceedings of the Third Central European Cybersecurity Conference. pp. 1–3 (2019)
21. Miller, C., Valasek, C.: Adventures in automotive networks and control units. Def Con **21**, 260–264 (2013)
22. Oruganti, P.S., Appel, M., Ahmed, Q.: Hardware-in-loop based automotive embedded systems cybersecurity evaluation testbed. In: Proceedings of the ACM Workshop on Automotive Cybersecurity. pp. 41–44 (2019)
23. Ritchey, R.W., Ammann, P.: Using model checking to analyze network vulnerabilities. In: Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000. pp. 156–165 (2000)
24. SAE Vehicle Electrical System Security Committee, et al.: Sae j3061-cybersecurity guidebook for cyber-physical automotive systems. SAE-Society of Automotive Engineers (2016)
25. Salfer, M., Schweppe, H., Eckert, C.: Efficient attack forest construction for automotive on-board networks. In: International Conference on Information Security. pp. 442–453 (2014)
26. Schneier, B.: Attack trees. Dr. Dobb's journal **24**(12), 21–29 (1999)
27. Sommer, F., Dürrwang, J.: Ieem-hska/aad: Automotive attack database (aad) (2019). https://doi.org/10.13140/RG.2.2.19528.37128, https://github.com/IEEM-HsKA/AAD
28. Sommer, F., Dürrwang, J., Kriesten, R.: Survey and classification of automotive security attacks. Information **10**(4), 148 (2019)
29. Thierry-Mieg, Y.: Symbolic model-checking using its-tools. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 231–237 (2015)
30. 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), https://unece.org/sites/default/files/2021-03/R155e.pdf