Enhancement of Automotive Penetration Testing with Threat Analyses Results

Article *in* SAE International Journal of Transportation Cybersecurity and Privacy · November 2018

DOI: 10.4271/11-01-02-0005

Jürgen Dürrwang
Karlsruhe University of Applied Sciences
22 PUBLICATIONS **208** CITATIONS
SEE PROFILE

Johannes Braun
Karlsruhe University of Applied Sciences
6 PUBLICATIONS **64** CITATIONS
SEE PROFILE

Marcel Rumez
Porsche AG
17 PUBLICATIONS **147** CITATIONS
SEE PROFILE

Reiner Kriesten
Karlsruhe University of Applied Sciences
54 PUBLICATIONS **398** CITATIONS
SEE PROFILE

Enhancement of Automotive Penetration Testing with Threat Analyses Results

Jürgen Dürrwang[1], Johannes Braun[1], Marcel Rumez[1], Reiner Kriesten[1] and Alexander Pretschner[2]

Institute of Energy Efficient Mobility (IEEM)[1], Karlsruhe University of Applied Sciences,
International University Campus 3, 76646 Bruchsal, Germany

Software Engineering[2], Technical University of Munich (TUM),
Boltzmannstr. 3, 85748 Garching, Germany

ABSTRACT

In this work, we present an approach to support penetration tests by combining safety and security analyses to enhance automotive security testing. Our approach includes a new way to combine safety and threat analyses to derive possible test cases. We reuse outcomes of a performed safety analysis as the input for a threat analysis. We show systematically how to derive test cases and we present the applicability of our approach by deriving and performing test cases for a penetration test of an automotive Electronic Control Unit (ECU). Therefore, we selected an airbag control unit due to its safety-critical functionality. During the penetration test, the selected control unit was installed on a test bench and we were able to successfully exploit a discovered vulnerability, causing the detonation of airbags.

INTRODUCTION

Modern automobiles host more than 50 ECUs, which contain a total of up to 100 million code lines to control safety-critical functionality [1]. In total, this fact and the close interconnectivity of automotive ECUs open up new possibilities to attack these systems. Many of these attacks affect the safe operation of the vehicle which can lead to injuries of the passengers [2][3][4][5]. To decrease the possibility of such scenarios, security has to be an integral part of the development lifecycle. In order to achieve the highest level of protection against security attacks, the Security by Design concept represents a promising attempt. Therefore, security tasks have to be embedded in early phases of the development lifecycle. One security-related task is the Threat Analysis and Risk Assessment (TARA). It identifies and prioritizes possible threats against the target which can lead to security incidents. Thus, countermeasures or design changes can be applied before the first line of code is written. In contrast, a validation of implemented security measures and a review of further vulnerabilities after an implementation should be performed. This can be done with a security testing of the system including internal and/or external testing. In particular, internal security testing is typically executed during the product development. On the other hand, external security tests, which are usually performed after product development, are called penetration tests if they are performed from the perspective of an attacker (see Figure 1, right path).

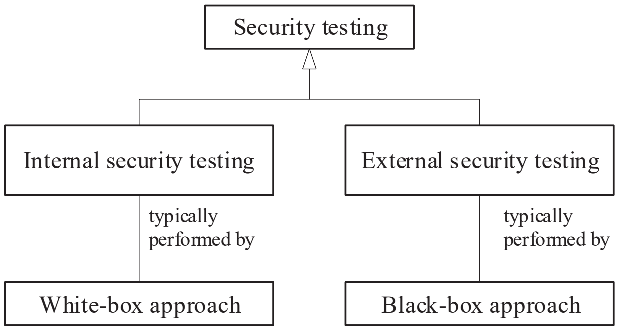

Figure 1: Classification of security testing and the supposed knowledge level

Internal security testing corresponds to a white-box approach having a great amount of information, e.g. development documents, source code, etc. Therefore, this testing scenario is best suited for validating of implemented security measures. To determine the resistance level of a system against hackers, penetration testing is the way to go [6]. Unfortunately, setting up a penetration test is not an easy task. Especially, if less knowledge of the device under test is available. Due to the fact that the tester has no detailed knowledge of the implementation but only a functional description of the system, the test can be described as a black-box test [7]. Hence, we consider it as reasonable to reuse the results of a threat modeling process to reduce the effort for security testing.

As safety violations can also be caused by security problems, it is important to analyse Hazard Analysis and Risk Assessment (HARA) results in terms of security. Beyond that, safety analyses performed with HARAs are a fixed part in the automotive sector. Results of these analyses contain valuable knowledge which can be used for threat modeling and therefore for penetration testing as well. Unfortunately, no approach exists which reuses HARA and TARA results for penetration testing. Consequently, we want to show the first step towards an approach that reuses results of both analysis types for decreasing the security testing effort. Moreover, we evaluate our approach by applying it for a safety critical ECU. In detail, the contributions of this paper are the following:

Problem: As penetration tests are mainly driven by tester-experience, an integration of threat analyses may increase the testing quality due to a complementary concept for test case generation.
Solution: We provide a methodology for test case generation with an adapted threat and risk analysis. To be more specific, we show the necessary steps for generating test cases using attack trees and how we embed these steps in a well-known security testing methodology. To do this, we transform identified threats into attack trees and derive test cases from each tree branch.
Contribution: Besides the derivation of test cases using attack trees, we also show how the results of a safety analysis can be reused. This will be done by the combination of a given Hazard analysis with a Threat and Risk analysis. In particular, we show how identified hazards can be used to identify threats that could compromise vehicle safety. In addition, we evaluated our methodology steps with a penetration test of an Airbag-ECU. As a result of our research, we have discovered a vulnerability that can cause life-threatening injuries to vehicle occupants and appears to exist across several manufacturers.

The paper is structured as follows: In the Section **Background**, we give a short overview of security testing concepts used in Information Technology (IT) and their suitability for the automotive domain. Furthermore, we summarize threat analysis methodologies which are known to be applicable for automotive systems. In the Section **Approach**, we show how to integrate threat modeling results for security testing. To show the practical applicability of our approach we provide an example in the Section **Experimental Evaluation**. Additionally, we suggest possible countermeasures to mitigate the discovered vulnerability, and conclude with an outlook in the Section **Conclusion and Future Work**.

BACKGROUND

SECURITY TESTING METHODOLOGIES A traditional approach in IT to assess the security of a system is the application of a penetration test. In this type of testing, the security engineers do not have any detailed information about the target system. The objective of penetration testing is to analyse the security level of a system, network or application from an attacker's perspective. This also implicates to use general public tools and methods which are available to a potential hacker. The primary goal of penetration testing is to find vulnerabilities that are not found during internal security testing and could be exploited by an adversary. Furthermore, penetration testing is usually a mandatory part of a security testing. The execution of penetration tests is normally at a late stage of the product development or afterwards. If any security issues are uncovered at this point, the costs and efforts of software patches are high in contrast to early development stages. Thus, a combination of white-box testing during the product development and penetration testing after the product release is reasonable.

For clarification, in this paper we don't want to propose a completely new penetration testing methodology, but rather an approach to identify feasible test cases for penetration testing by using threat modeling techniques. Therefore, we have analysed publicly available security testing methodologies, which are commonly applied in the traditional IT. In contrast, there are still no guidelines explicitly for the automotive sector. The Open Source Security Testing Methodology Manual (OSSTMM) [8] includes fundamentals in security risks, metrics, and disclosure as well as planning and execution of different test types. However, there is no part with detailed information with a focus on penetration testing. The special publication 800-115 of National Institute of Standards and Technology (NIST) [9] gives a good overview and practical recommendations for security testing similar to OSSTMM without the intention to represent a comprehensive security testing guideline. In contrast to OSSTMM, the Information Systems Security Assessment Framework (ISSAF) [10] is very comprehensive and presents strategies, assessments, and check-lists. Beyond that, ISSAF includes also a penetration testing methodology and defines different steps for performing these type of test but does not integrate a threat analysis in its steps. In addition, the structure as a framework allows an integration in the business life cycle [11]. Furthermore, the Penetration Testing Execution

Standard (PTES) [12] is a technical guideline exclusively for penetration testing. Similar to ISSAF, it contains different steps to perform a comprehensive test procedure and explains each step in detail with recommendations, which software tools can be used. Furthermore, the PTES recommends to perform a threat modeling step as the only method. Unfortunately, the standard does not suggest a specific model but rather which consistent terms should be supported by the used model. The guideline presented by the Open Web Application Security Project (OWASP) [13] can be seen as a special field for testing the security of web applications.

In summary, the analysed security testing methodologies have different characteristics relating to penetration testing. Only the PTES guideline supports the approach of threat modeling, which is fundamental part of our approach to derive test cases. As we mentioned before, PTES does not recommend a specific methodology for this step. For this reason, we depict differences between existing threat modeling methodologies in the next section relating to their applicability for Cyber-Physical-Systems (CPS) due to their special relationship between safety and security.

THREAT MODELING The concept of threat modeling is an integral part of our overall methodology and therefore we will briefly describe the existing methods for threat modeling of automotive systems in the following. We want to start with approaches explicitly recommended for the automotive sector and listed by SAE in its Cybersecurity Guidebook for Cyber-Physical Vehicle Systems (SAE J3061 [14]). The guidebook suggests to apply the Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE) approach, the HEAling Vulnerabilities to ENhance Software Security and Safety (HEAVENS) [15] methodology, and the E-Safety Vehicle Intrusion Protected Applications (EVITA) [16][17] approach. OCTAVE was developed for enterprise security risk assessment. Therefore, it does not directly address specific attributes of automotive systems which makes it less meaningful to apply in the automotive domain. The HEAVENS methodology is focused on threats of the Microsoft Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege (STRIDE) classification model [18][19]. Due to the fact that the scope of STRIDE is mainly on security goals, it depends on the security engineer to decide whether a threat violates the safety of a vehicle or not. This fact implies that security engineers have to be familiar with safety to decide if a threat can violate a safety goal. Lastly, EVITA uses dark-side scenarios as unintended use cases to identify possible security threats. The methodology is mainly focused on vehicle-to-vehicle connections and less on in-vehicular networks. The threat identification depends highly on the set of use cases (dark-side scenarios) whereas safety-related use cases are less in focus. As a result, threats with impact on safety can be unconsidered by using this approach. Furthermore, the classification of safety and non-safety threats is different. Hence, this can lead to an unbalanced threat assessment [20].

Besides the listed threat modeling methodologies in the Society of Automotive Engineers (SAE) guidebook, we want to mention the Hybrid Threat Modeling Method (hTMM) approach developed by a CERT division [21]. The approach combines Security Cards [22] and the persona non grata (PnG) for threat and risk identification. Especially, the provision of so-called Security Cards to guide the analyst through the brainstorming phase should be mentioned. Besides its benefit to support the analyst in the brainstorming phase, the approach does not directly focus on identification of threats that can violate the safety of the item under analysis. In conclusion, the approach is not developed to import results of a pre-performed safety analysis.

Threat Modeling Including Safety We were particularly interested in identifying test cases that can violate the safety of the vehicle. Therefore, a threat modeling methodology should be able to identify this particular threat type. To obtain a considerable number of candidates for such threat modeling methodologies, we conducted a literature survey and identified additional threat analysis methodologies which were not explicitly mentioned in the SAE guidebook. In particular, we additionally found the Security-aware Hazard and Risk Analysis Method (SAHARA) [23][24] and the Failure Mode Vulnerabilities and Effect Analysis (FMVEA) [25][26]. We decided to assign the identified methodologies to the classes of safety and security depending on their focus and according to our point of view. The resulting classification contains exclusive methods for HARA and TARA as well as a categorization for combined methodologies like HEAVENS or CORAS [27] shown in Figure 2.

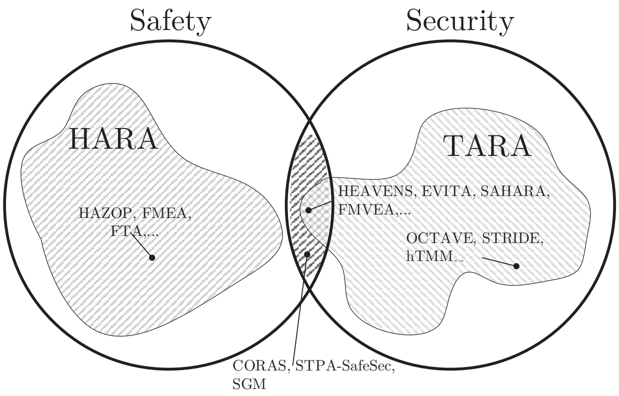

Figure 2: Classification of HARA and TARA methods which were selected from the literature survey

Table 1: Threat analysis methodologies with regard to safety, mentioned in the automotive context

Criteria	SAHARA	FMVEA	EVITA	HEAVENS
Scope of application	Automotive systems	Cyber-Physical-Systems (CPS)	Automotive systems	Automotive systems
Threat modeling approach	STRIDE	STRIDE	Dark-side scenarios with attack trees	STRIDE
Type of safety & security linking	Hazards and threats are identified and security threats with safety consequences are passed on to safety analysts.	Failure modes and threats are detected in parallel without any exchange between the both domains.	Only risk assessment of threats in terms of safety.	Only risk assessment of threats in terms of safety.

HARA is a subset in the safety set shown in Figure 2 illustrating methods which are exclusively used for safety analyses. In particular, these are the Hazard and Operability Study (HAZOP), the Failure Mode and Effects Analysis (FMEA) or the Fault Tree Analysis (FTA). On the other hand, TARA methods are a subset of the security set. We made a distinction between threat modeling methods that do not address safety (OCTAVE, STRIDE and hTMM) and methodologies that address safety in specific parts, e.g. determining severity values for threats violating safety. These are in particular: HEAVENS, EVITA, SAHARA and FMVEA. Besides the distinction, if a proposed threat modeling methodology addresses safety, it was of interest at which point this is done. Therefore, we evaluated the threat modeling methodologies listed in Figure 2 with the criteria in Table 1. It was interesting to know, on which domain the proposed methods are focussed, the technique for identifying threats and especially if the methodology provides a link between safety and security.

Our analysis has shown that EVITA and HEAVENS only address safety after security threats have been identified. This is done by a safety risk value if the threat influences safety. This means that these methodologies do not explicitly focus on the threat identification regarding safety. Furthermore, both methodologies do not integrate or claim a safety analysis in their description. However, SAHARA and FMVEA have a stronger focus on safety. Both methodologies determine risk values for security threats which have a safety impact. Besides this, SAHARA claims to perform a HARA as well as a TARA in a parallel manner. In addition, the methodology examines if the identified threats can violate the safety of the vehicle and forwards identified threats to safety analysts who rate the safety impact [24]. Unfortunately, neither SAHARA nor FMVEA reuse identified failures or hazards of a safety analysis as an input for their threat analysis. This implies that the outcomes of a performed safety analysis are not used for threat identification. Thus, it depends on the security engineer to decide if a threat is safety-critical or not. As a result, security engineers must have safety knowledge.

Methodologies from other domains like Information and Communication Technology (ICT) provide the capability to reuse results of a perform safety analysis. A holistic overview of such approaches is given by the researchers Friedberg et al. [28]. They provide a survey of research papers for combined safety and security analyses which can be selected for CPSs. Friedberg et al. list different approaches classified in two categories: generic and model-based approaches. The latter are subdivided into graphical and non-graphical approaches. As part of the evaluated methods we want to mention the CORAS [27] approach and STPA-sec [29] with the extension STPA-SafeSec [30]. These methodologies suggest performing a hazard analysis followed by a security threat analysis. This makes it possible to include and reuse results of safety analyses. Furthermore, the methods can be applied in an early development phase which offers advantages for early design decisions. For the analysis, CORAS uses UML models and STPA-SafeSec use control loops with component layer diagrams to identify security constraints [30]. Both approaches meet most of our demands for a combined safety and security method. But they are rather complex and not easy to integrate into existent safety analyses due to the fact that they require specific method steps. As a result, we decided to use another approach for a combined safety and security analysis that can be easily integrated into existing safety analyses while also achieving a link between safety and security. The method is called Security Guide-word Method (SGM) and will be described in the next section.

Threat Modeling Using the Security Guide-word Method

In this section, we give an overview of our previous presented approach [31] for combining safety and security analyses by reason of reusing this method as a component for our security testing approach. The approach reuses identified hazards as an input for the threat analysis without changing the typical safety analysis steps (see Figure 4) and can be easily integrated into established safety and security processes presented in Figure 3. We recommend our method to safety engineers as it enables them to identify threats that can impair vehicle safety.

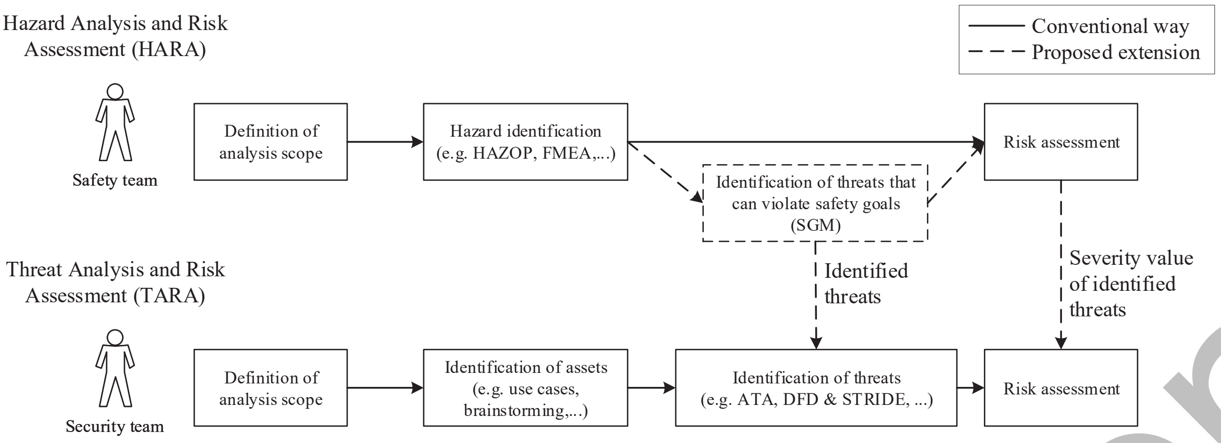
Hazard Analysis and Risk Assessment (HARA)
Conventional way
Proposed extension
Safety team
Definition of analysis scope
Hazard identification (e.g. HAZOP, FMEA,...)
Identification of threats that can violate safety goals (SGM)
Risk assessment
Identified threats
Severity value of identified threats
Threat Analysis and Risk Assessment (TARA)
Security team
Definition of analysis scope
Identification of assets (e.g. use cases, brainstorming,...)
Identification of threats (e.g. ATA, DFD & STRIDE, ...)
Risk assessment

Figure 3: Common steps for safety and security analyses with the SGM extension to reuse safety results

For this purpose, we extend the typical safety analysis process with our approach as an intermediate step, which is shown in Figure 3 by the dashed box. The analyst is guided through the brainstorming phase using guide-words and a template (see Table 2) to achieve a structured threat identification. The use of guide-words was originally introduced by HAZOP for functional safety analysis and in compliance with ISO 26262. HAZOP was then extended by the researchers Rune et al. [32][33] for security, whereupon we extended the approach to reuse the results of a safety analysis to identify threats. In particular, we derive threats from the identified hazards violating the safety of the vehicle. We call this method the SGM and we demonstrated the applicability of the method by means of an experimental evaluation with safety and security engineers [31].

We want to point out that SGM is actually only focused on security threats which can violate safety goals. We consider this as reasonable due to the negative consequences that safety violations can cause. Furthermore, manufacturers are committed to performing a safety analysis and therefore they should look for security threats which influence safety goals. Currently, assets like confidentiality or repudiation are not directly addressed and therefore they should be addressed by using one of the listed threat methodologies above, e.g. STRIDE or hTMM.

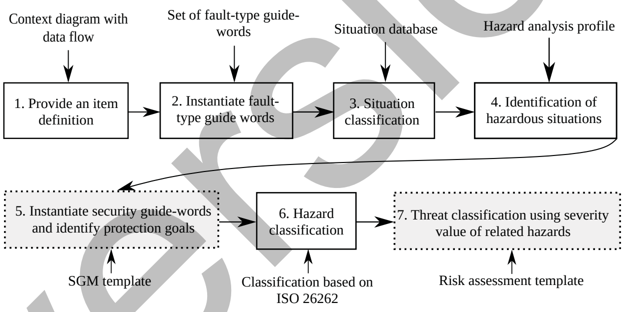
Context diagram with data flow
Set of fault-type guide-words
Situation database
Hazard analysis profile
1. Provide an item definition
2. Instantiate fault-type guide words
3. Situation classification
4. Identification of hazardous situations
5. Instantiate security guide-words and identify protection goals
6. Hazard classification
7. Threat classification using severity value of related hazards
SGM template
Classification based on ISO 26262
Risk assessment template

Figure 4: General safety analysis steps according to ISO 26262 complemented with the steps for the SGM [31] which is represented by the dashed boxes.

To illustrate the embedding of our SGM method in the established process of a safety analysis according to ISO 26262, the individual steps are shown in Figure 4 whereas Step 5 presents an additional step based on our SGM approach. The method receives hazards as input, which were identified in the steps before. To systematically derive threats that can violate safety goals, we use each identified hazard with a set of guide-words, done as in HAZOP. For this, we provide a template which supports the structured threat analysis, shown in Table 2. After finishing this step, the safety analysis continues with determining risk values for each hazard, presented by step 6 in Figure 4. Lastly, we take over the severity values of step 6 and use them as one parameter for determining risk values for threats identified with SGM.

Table 2: SGM template.

Hazard	can be triggered by	name of message or function	for component or system	entry point
(1)	(2)	(3)	(4)	(5)

The template in Table 2 requires a hazard in column 1 which was identified in the safety analysis (HARA). In column 2 one security guide-word is required from the set: *disclosure of information*, *disconnection of ECU*, *delay of message or signal*, *deletion of message or signal*, *stopping of ECU*, *triggering of (diagnostic) function*, *insertion of message or signal*, *reset of ECU*, *manipulation of message or signal* and lastly *manipulation of firmware*.

Furthermore, in column 3 the analyst writes down the name of the message or function which will be the objective of the security incident. To obtain more information about the identified threat, column 4 and 5 are implemented in the template. Column 4 represents the name of the component or system that will potentially be under attack. The last column in the given template records the point of entry, where the attack could potentially start. As an example, for the manipulation of a message transmitted over Controller Area Network (CAN), the attacker needs access to CAN. Hence, the analyst would write down *CAN* in column 5. We consider it as reasonable that safety analysts can generate a great contribution to threat modeling by applying SGM. This is partly explained by the fact that safety analysts obtained a lot of knowledge during their safety analysis that can be reused for security, e.g. the item definition or high-level architecture. This hypothesis seems to be confirmed by an evaluation we performed with safety and security engineers [31][34]. As a last step of the HARA based threat modeling part, we suggest transferring the identified threats to security engineers for further evaluation which can be seen in the lower part of Figure 3.

APPROACH

As we have already explained, the suggested approach can be used in the scenario of internal and external testing related to Figure 1. In this work, we will explain the steps of our approach for external testing and therefore as a penetration testing scenario. This can be explained by the fact that the external and black-box scenario is typically defined for no or little knowledge about the device under test. Consequently, this scenario represents the more difficult scenario of both.

The main goal of our approach is to provide a guideline for security testers and if possible to reduce the testing effort for the security testing by reusing existing analysis results. The steps suggested in Figure 5 are in accordance with the PTES standard, as we consider this penetration testing methodology as suitable for automotive systems by including a threat modeling step exclusively. Therefore, the upper part of Figure 5 show the original steps of PTES and the lower part represents the steps we modified. As with PTES, Step 1 is representing the intelligence gathering phase, we recommend collecting as much knowledge about the device under test as possible. Step 2 then recommends performing a hazard and threat analysis as described in Section **Threat Modeling Using the Security Guideword Method**, to identify security aspects which should be tested. We would like to point out that the execution of a HARA and TARA in Step 2 can be considered as optional if they were already performed in an early product development phase (internal testing scenario). In contrast, this means that additional effort is only generated in the external testing scenario. However, we consider this effort to be very helpful, as it allows the tester to gain a better understanding of the black-box device.

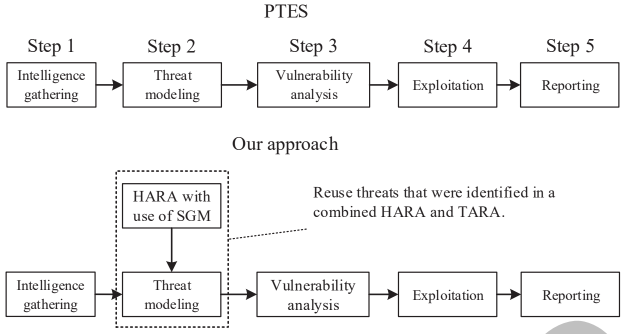

Figure 5: Method steps for deriving test cases with hazard and threat analyses in addition to the PTES phases

It was crucial that we did not change the original steps of PTES but adjust them for integration of our approach. Thus, the following sections will show how our approach can be integrated into the phases *Intelligence Gathering* and *Threat Modeling* of the PTES methodology. In addition, Figure 6 shows the sequence and the required subtasks from Step 2 out of Figure 5. The generic sequence starts with a safety analysis and ends with the derivation of test cases based on the threat modeling results represented by Step 2.4.

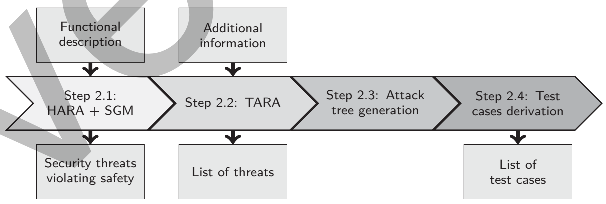

Figure 6: Detailed description of Step 2 of Figure 5 which shows the sequence for deriving test cases based on HARA and TARA results using attack trees.

As a first step, we recommend performing a safety analysis (HARA) with SGM shown with Step 2.1 in Figure 6. The input is a functional description of the device under test. As output, we receive a list of security threats which can violate the safety of the vehicle. Based on this a threat analysis in Step 2.2 is performed. In particular, threats identified with SGM represent the class of threats that can violate the safety of the vehicle. Additional information about the device under test should be added to this step. In particular, all relevant information gathered in the intelligence gathering phase which will be explained in the following.

For common threat classification, we additionally provide a mapping between the SGM and STRIDE terminology [18] in the Appendix. With regard to the terminology of an unintentional event, we use the expression *threat*, which can cause an attack to trigger an undesired event.

INTELLIGENCE GATHERING The information gathering phase is the first and most important step of the whole method. All further steps are based on this. The objective of the phase is to collect as much knowledge of the device under test as possible. Due to the fact that we have focussed on the external test scenario in this paper, we want to point out that this step in the internal test scenario requires the collection and analysis of existing development documents in regard to security. However, in the black-box scenario, this phase involves collecting information from public sources like product, technical or functional descriptions. In addition to these sources of information, international standards also provide useful details about the test candidate for the derivation of test cases. This applies in particular to the automotive sector by means of the extensive standardization of electronic components. Additionally, we consider it as reasonable to further focus on the following points:

- Is there a high-level architecture of the device under test available?
- Can we determine a general overview of the communication of the device under test?
- Does the device under test provide diagnostic functionality?
- Are there external interfaces, e.g. On-board Diagnostics (OBD)-Port, connected with the device under test?

Thus, it is often possible to reconstruct a high-level architecture of the device under test. IIn particular, how the device is embedded in bus systems and how it is connected to actuators or other electronic components. At first glance, this appears to be a challenge, but automotive architectures usually differ only slightly from one to next generation. Therefore, previous architectures can be often reused to gather the desired information. Moreover, we consider it as reasonable to identify how the item communicates with its environment. Therefore, tools which are able to record or send bus messages can be used [35]. In addition, scanning the environment for possible communication partners opens up the possibility to specifically manipulate the device under test. For example, this can be done with a port scanner [36]. Beyond that analysing and simulating of specified communication is an important step to get an ECU in the normal operation mode, due to the fact that ECUs typically do not operate, if the expected communication behaviour is not available. Furthermore, it makes sense to check whether the ECU possesses diagnostic functionality. If this is the case a high probability exists that the ECU can be accessed over the OBD-Port by sending diagnostic messages. From our point of view, the OBD-Port is usually a good starting point for additional information gathering.

THREAT MODELING USING HARA AND TARA In addition to the publicly available information sources mentioned before, the outcome of a performed threat analysis increase the knowledge of the device under test. Whereas in the internal testing scenario, a threat analysis is carried out in an early stage of the product development phase, in the external scenario it is explicitly performed to increase the knowledge of the device that is presenting itself as a black-box to the tester. To be more precise, the first option results in the highest knowledge level due to the fact that a threat analysis in an early product development phase is done by the manufacturer with a deep understanding of the device under test. In the second scenario, a safety and threat analysis is exclusively performed by the external tester to increase the knowledge level of the device under test. Therefore, information collected in the intelligence gathering phase is a valuable input for the safety analysis followed by a threat analysis. As an example, the functional description derived from publicly available documents is a useful input.

ATTACK TREE GENERATION In Step 2.3 of Figure 6, we perform each identified threat to create attack trees. Dependent on the type of the selected threat modeling method (STRIDE, attack trees, etc.), attack trees can be taken over directly or have to be generated in this step. In the case of creating attack trees, we want to present a general attack tree in Figure 7. The root of the tree represents the threat identified in Step 2.2 and each node below represents a conceivable intermediate attack step.

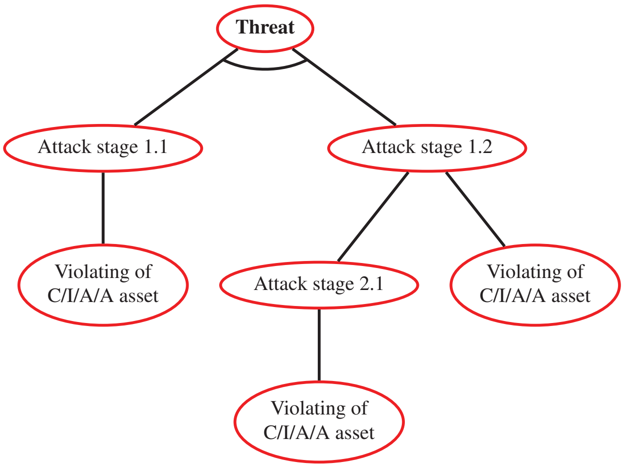
Threat
Attack stage 1.1
Attack stage 1.2
Violating of C/I/A/A asset
Attack stage 2.1
Violating of C/I/A/A asset
Violating of C/I/A/A asset

Figure 7: General attack tree with one root, three intermediate nodes, and three leaf nodes. Whereby the first two nodes have an *AND* conjunction and the other nodes an *OR* conjunction. Leaf nodes are described with the Confidentiality, Integrity, Availability and Authenticity (CIA(A)) nomenclature.

The presented tree uses the logical conjunctions *OR* and *AND* to describe the conditions which have to be met. Whereby the conjunction *AND* is represented by a semicircle between all nodes which have to be successfully exploited at the same time to reach the parent node. This is obviously done with the nodes *Attack Stage 1.1* and *Attack Stage 1.2* in Figure 7. The other nodes in the tree have an *OR* conjunction which means only one node has to be successfully exploited to reach the parent. By this, the assumed feasibility of an attack can be determined through its child nodes. Therefore, a scenario linked to the requirements with an *OR* is generally easier to achieve then an *AND* conjunction. We would like to emphasize that each branch of the tree in Figure 7 ends with a leaf node. These nodes are representing the violated information assets related to the CIA(A) triad and will further support the derivation of test cases which will be explained in the next section.

DERIVATION OF TEST CASES With the results of Step 2.3 in Figure 6, we can start to derive test cases. Therefore, we use each branch of the created attack tree. In particular, we follow the selected path starting in the root element and ending in a leaf node. To be more precise, for each attack stage we are able to specify the test case related to the violated information assets represented by the leaf node. As an example, we perform the left branch of the attack tree in Figure 9. The threat *Unintended airbag deployment* can be initiated with the attack vector *Triggering of function airbag deployment*. Broken down to the CIA(A) assets this is equal to a violation of the authenticity and integrity of a diagnostic message. Based on this information, we can derive the test case: *Try to violate the authenticity and integrity of the diagnostic message used for airbag deployment*. In this way, we use each path in the tree and derive the corresponding test case for each intermediate node based on its leaf node.

EXPERIMENTAL EVALUATION

We evaluated our approach by a penetration test for a Pyrotechnic Control Unit (PCU), which controls the pyrotechnic charges in a vehicle, e.g. airbags, battery clamps, belt tensioners, etc. We chose this control unit due to the fact that to this time no attack against an Airbag-ECU was known. In addition, we have already been successful on other ECUs by manipulating safety-relevant control devices such as the engine-ECU or the brakes. Moreover, we passed through each step of our approach beginning with the intelligence gathering part followed by a threat modeling with the outcomes of a performed HARA. The penetration test was performed without knowledge of the source code or the CAN communication matrix, which reflects our mentioned division into a black-box scenario.

INTELLIGENCE GATHERING OF THE PCU As described in the approach before, we started with the intelligence gathering as the first phase of the penetration testing procedure. Therefore, we read up on the operating mode of the Airbag-ECU (also referred to as PCUs) and how this functionality is typically implemented to get a comprehensive understanding of PCUs. However, the found sources were mostly from a technical and safety point of view, lacking the discussion of integrated security measures. From the point of view of a penetration tester, the general structure of a PCU proved to be very challenging to discover a vulnerability. By relying on multiple signal sources, e.g. hard-wired sensors, integrated sensor in the PCU itself and information acquired from bus messages, manipulating signals able to pass a plausibility check was considered to be very difficult. Moreover, without the availability of the source code, the logical conjunctions of these plausibility checks represent an additional challenge for the tester. During the study of International Organization for Standardization (ISO) standards 14229 [37] and 26021 [38], it was discovered, that some PCUs support a End-of-life (EOL) deployment based on Unified Diagnostic Services (UDS) [37], to aid in the recycling process of automobiles. Although standards are not available free-of-charge, they can be bought from the ISO in digital or printed form.

High-Level Architecture of the PCU After analysing the publicly available information, we reconstructed the connectivity of the target. This overview and the later performed threat analysis helped us in identifying possible *attack vectors*. As shown in Figure 8, the component is connected to an automotive bus system via an end-to-end connection.

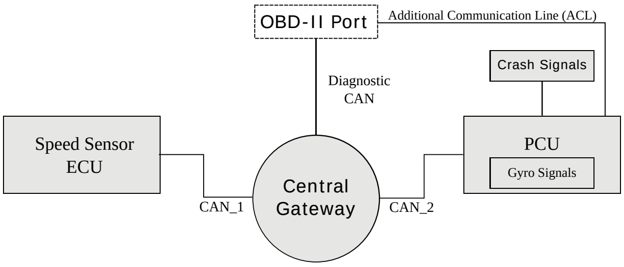

Figure 8: Assumed high-level architecture of device under test

Furthermore, it shows the device under test (PCU) is connected via CAN to the Central Gateway (CGW). There are two different internal signal paths for communicating with the PCU either over the CGW or directly by physical access to the *CAN_2* wires between CGW and PCU. In addition, an external communication can be established via the OBD-Port by using the *Diagnostic CAN*, which is connected to the gateway. Furthermore, the PCU has two additional connections, one for receiving data from crash sensors (*Crash Signals*) as well as the Additional Communication Line (ACL). The ACL is a link between

PCU and OBD-Port. The connection is intended as an additional communication line in besides to the CAN communication. Manufacturers can use this additional communication to adapt the EOL detonation process to their needs [38]. To clarify, both connections are hard-wired in the physical sense, but we are going to refer to the end-to-end connections as hard-wired signals in this work.

THREAT MODELING FOR THE PCU We performed a safety analysis for the selected device (PCU) and its definition, shown in Figure 8. As a result, we identified several hazards with the HAZOP approach and ranked *Unintended airbag deployment* as well as *Unintended prevention of airbag deployment* as the most critical hazards. After a discussion with safety engineers, we decided to focus on *Unintended airbag deployment* due to its high probability to injure passengers. Consequently, we applied the SGM approach as suggested in Figure 6 and were able to identify several threats, which could probably cause a deployment of charges. For reason of presentation, we present only the following threats and how we used them further.

1. Unintended airbag deployment can be triggered by triggering of (diagnostic) function *airbag deployment*.

2. Unintended airbag deployment can be triggered by manipulation of message *diagnostic message*.

3. Unintended airbag deployment can be triggered by manipulation of message *crash message*.

The first threat describes an unauthorized activation of a standard or diagnostic function of the PCU. For clarification, a standard function, in our view, is a function which is implemented in a way to run during normal operation, e.g. triggering airbags after an accident. The other type of function (diagnostic function) is implemented to support technicians in workshops to solve problems during repairs. This type of function has shown a high probability to get exploited in the last years [39]. Besides this, Threat 1 has a high probability to be existent in our device under test, since almost all ECUs controlling actuators possess functions for self-diagnostics. Besides the unauthorized activation of a diagnostic function, it is possible to manipulate a message which is used by a diagnostic function. This case is represented by Threat 2 and can result in an undefined behaviour leading to airbag detonation as well. Lastly, Threat 3 describes the manipulation of a message which contains the information about accident detection. For example, such messages are typically sent by intelligent crash sensors to the Airbag-ECU.

ATTACK TREE GENERATION FOR THE PCU As proposed by our approach, we generate attack trees for the identified threats. Therefore, we take the first part of the threat as the root of our attack tree shown in Figure 9. In particular, for Threat 1 this is *Unintended airbag deployment*. Then, we use the second part of the threat description to create the next lower level in the tree below the root. Lastly, we define the leaves of the tree by the information assets that are violated. To do so, we use column 2 of Table 4, filled out with the information assets in relation to each SGM guide-word. For Threat 1 and guide-word *triggering*, we take over the information assets authenticity and integrity.

DERIVATION OF TEST CASES FOR THE PCU For the derivation of the test cases, we used the tree shown in Figure 9. Here the selected threat and the different attack levels executed by an attacker can be derived. Furthermore, the relevant information assets are identified, i.e. the authenticity and integrity of diagnostic messages or the integrity of environmental perception of a crash message for the airbag system. For faking an accident situation, we assumed the manipulation of sensor data represented by the upper right path in the attack tree. However, this means that all three signal sources must be manipulated simultaneously. We assumed this preconditions as essential due to the fact that in PCUs plausibility checks are implemented to prevent an unintended airbag deployment by faulty signal sources. This assumption was additionally supported by a common practice of integrating plausibility checks in functional safety concepts.

Consequently, we used each path in the attack tree for the derivation of individual test cases. In particular, as a first test case, we derived: *Try to violate the authenticity and integrity of the diagnostic message used for airbag deployment*. One example for this is the replay of a message, which was recorded during a diagnostic session between PCU and diagnostic tester. Performing the second path of the tree, we derived: *Try to manipulate the payload of messages which are used in the diagnostic session*. For the manipulation of payloads, we simply tried all possible combinations on binary level. Lastly, the third path in the tree leads to the test case: *Try to fake crash situation by manipulating crash signals or message*. As we knew that for this case we would have to manipulate three different signal sources, we concluded that this path should be the most difficult one.

Regarding the question which test case should be performed first, we ranked them by the length of each branch in the tree and the number of information assets to be violated. Hence, we started with the first branch in the attack tree, followed by the second and third branch in Figure 9.

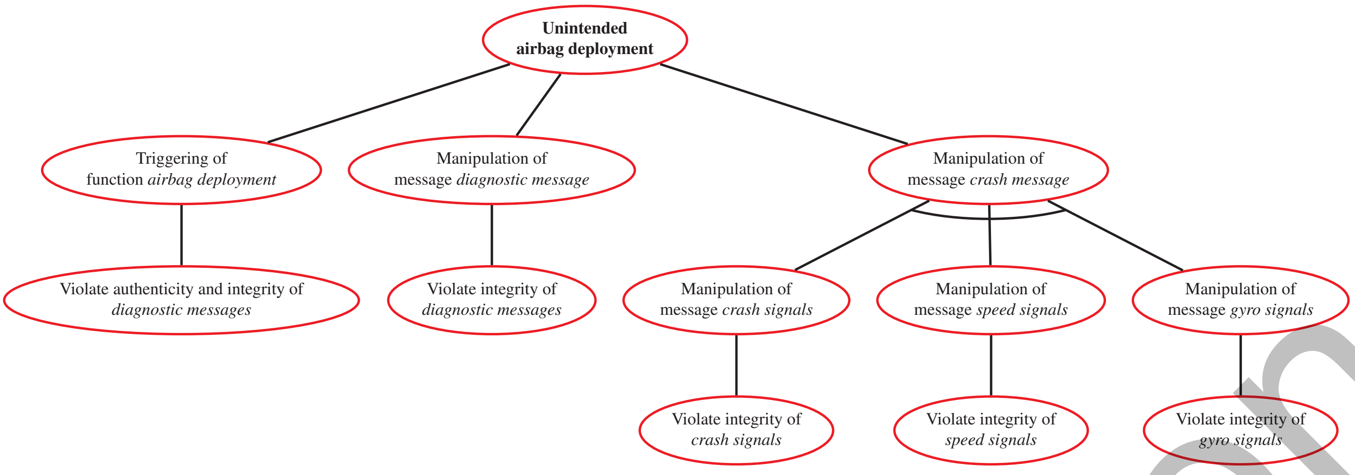
Unintended airbag deployment
Triggering of function *airbag deployment*
Manipulation of message *diagnostic message*
Manipulation of message *crash message*
Violate authenticity and integrity of *diagnostic messages*
Violate integrity of *diagnostic messages*
Manipulation of message *crash signals*
Manipulation of message *speed signals*
Manipulation of message *gyro signals*
Violate integrity of *crash signals*
Violate integrity of *speed signals*
Violate integrity of *gyro signals*

Figure 9: Attack tree for the threat "‘*Unintended airbag deployment*"’ using SGM nomenclature

EXPLOITING OF IDENTIFIED VULNERABILITIES

In this section, we want to show how we performed the selected test cases in our penetration test. Due to the length of this paper, we will show the procedure using the threat *Unintended airbag deployment can be triggered by triggering of (diagnostic) function airbag deployment* as an example. Hereby the threat is represented by the left branch of the attack tree in Figure 9. The related test case for this threat is *Try to violate the authenticity and integrity of the diagnostic message used for airbag deployment*. In combination with the knowledge collected in the intelligence gathering phase and the reference to the EOL standard, we decided on trying to gain unauthorized access to diagnostic based EOL functionality.

Identification of Vulnerable Vehicles In a first step, we had to check the implementation of the EOL capability according to the ISO standard. Therefore, a diagnostic scan (UDS) on the test vehicles was performed. The standard mandates that the first Airbag-ECU inside a vehicle listens and replies to diagnostic requests (UDS) with the CAN ID shown in Table 3. This is the so-called fixed-address and enables us to directly identify if an Airbag-ECU (PCU) has implemented the relevant EOL standard.

Table 3: CAN Identifiers (IDs) for communication with fixed-address PCU regarding the airbag standard [38]

	11 bit CAN ID	**29 bit** CAN ID
Request	0x7F1	0x18DA53F1
Response	0x7F9	0x18DAF153

In the case that a vehicle contains more than one airbag unit, the CAN IDs of the other units can be requested via a diagnostic service reached from the first Airbag-ECU (fixed-address PCU). If a PCU does not respond to the sent messages, it can be assumed directly that the EOL standard has not been implemented.

Test Bench for Penetration Testing After the EOL functionality in a vehicle was discovered, we purchased the built-in PCU as a replacement part and developed a test bench. We did not want to launch a penetration test on a real vehicle, as unintentional deployment of airbags can be very dangerous. The test bench is shown in Figure 10.

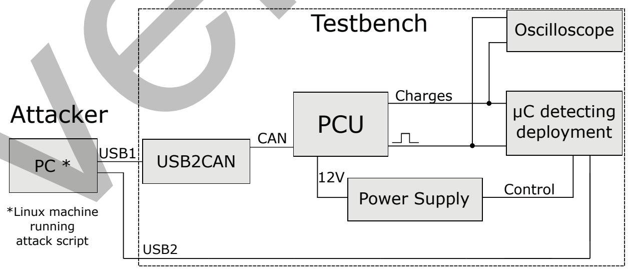

Figure 10: Schematic layout of the developed testbench

The setup consists the PCU, which is connected to a Linux machine via a USB-to-CAN adapter. The Linux machine runs the scripts to perform the penetration test and was additionally used for documentation purposes. Furthermore, we connected a microcontroller (interrupt based) and an oscilloscope to all outputs pins of the PCU to detect the voltage pulse which triggers the charges to detonate. This setup enabled us an independent and automatic execution of test cases without the need for a constant supervision.

Exploitation For our penetration test, we performed the above-mentioned test case. In particular, an unauthorized activation of the diagnostic functionality for the end of life detonation. To do so, we used the information given by the EOL standard, which describes how a tool, called Pyrotechnic Device Deployment Tool (PDT), must be implemented to allow recycling companies the deployment of pyrotechnic charges inside of compatible vehicles. As the standard describes deployment via CAN only and other methods including an additional wired connection (ACL), a target without that additional connection was selected.

Our way to exploit the selected PCU is described by seven steps and is shown in Figure 11. The steps are based on the deployment of airbags via UDS which can be extracted from the ISO 26021 standard.

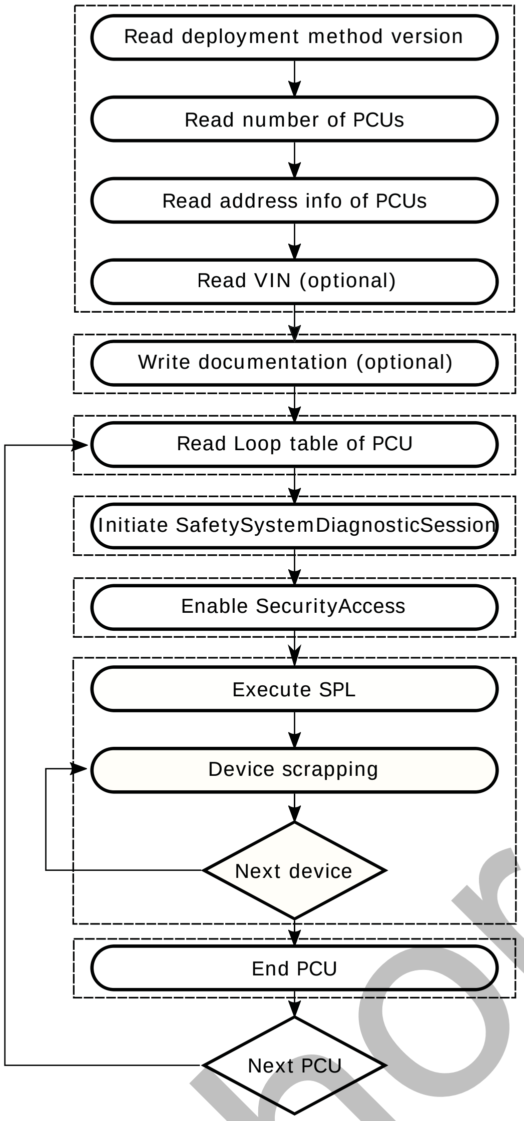

Figure 11: The steps performed during the penetration test, based on the flowchart according to the ISO standard 26021 [38]

- In Step 1, we used the *Read Data By Identifier* service to retrieve information from the target. This provided us with valuable information, e.g. the amount of PCUs inside the vehicle. It shall be noted, that none of the vehicles tested during the research phase was using more than one PCU.

- In Step 2, we would have been able to use the *Write Data By Identifier* service to store information, e.g. the date of deployment inside the PCU or data about the individual performing the recycling deployment. With our penetration tests simulating a malicious attack, we decided to not use this function, as it is highly unlikely that someone would want to leave intentional traces of a malicious deployment. Moreover, this step proved to be unnecessary for the execution of the next steps.

- In Step 3, we used the *Read Data By Identifier* service in the same way as described in Step 1. However, the response to this request was usually the longest, as it does not only include the type of all *loops* but their current status as well. Besides helping us in setting up the attack this enabled us to gather additional information during the whole experiment, as we could determine what changes to the test-bench made the internal status codes switch.

- Step 4, was our first challenge, as we had to use the *Diagnostic Session Control* service to unlock enhanced diagnostic functionality by switching into a *Safety System Diagnostic Session*. The target requires certain conditions to be met, in order to allow switching into this diagnostic session. These conditions were not immediately clear. While certain Negative Response Codes (NRCs) [37] are helpful, e.g. *RPM too high*, it was observed that most of the time PCUs used the NRC *conditions not correct*, leaving us with no hint what may have caused the rejection of the request. By changing the physical signals and bus simulation playback according to the NRC combined with scenarios that made sense for the vehicle state during a recycling deployment we found working boundary conditions. Based on those we were able to further modify the boundary conditions to find the most dangerous, although working, combination of vehicle state values. Contrary to the *Default Diagnostic Session*, this session requires the cyclic transmission of a *Diagnostic Tester Present* message in order to prevent a timeout.

- In Step 5, the *Security Access* service protected the access to specific functions. To ensure that a diagnostic tester is authorized to access these functions, it uses a *challenge-response* procedure. Because we had no knowledge of the used algorithm we acquired multiple seeds for analysis and implemented the possibility to enter the key by hand or with an automatic calculation. After determining the correct algorithm, the calculation was implemented in our tool to automate the attack as much as possible.

- In Step 6, we were able to use the *Routine Control* service to execute the Scrapping Program Module Loader (SPL). The SPL makes the Scrapping Program Module (SPM) executable by converting it in executable code and moving it into the PCU's RAM. Without these two steps a deployment using this attack vector is not possible. After successfully performing these preparations we used the *Routine Control* service for the third time to deploy a specific charge. This specific charge is selected via sending the *loop ID* in the 5th byte of the *Routine Control* message. In case more than one pyrotechnic charge is available for deployment, the message can be sent again with the new *loop* ID in place. During our experiment, we decided to deploy a single loop and monitor the PCU's output and internal status values.

- In Step 7, we ended the scrapping of attached charges by using the *EcuReset* service to *hardreset* the PCU. In this case, performing a *hardreset* power cycled the PCU, leading to the loss of data stored inside the Random-access Memory (RAM). Thus, stopping the ability to deploy charges, as the necessary code is only executable after performing the required conversion steps in the on-state as described in Step 6.

PENETRATION TEST RESULTS Related to Step 5 in Figure 5 we reported the vulnerabilities, which were discovered during the penetration test. We discovered issues exclusively based on the software implementation, e.g. the weak security access and we discovered issues where the weakness consists of an unused combination of hard- and software.

Vulnerability in Security Access With the *Security Access* service having a history of being exploited, this access was assumed to possess a vulnerability. In particular, the algorithms analysed so far violate the *Kerckhoff's Principle* [40] by using the algorithm as secret instead of the key. In this case, we analysed multiple seeds to check for patterns and the overall amount of different seeds provided by the target. Here, the given target provided new *seeds* for different requests and after the target has been power cycled.

While studying the EOL standard, we discovered the description of the implementation for the *Security Access* service to protect against unauthorized deployment. This is followed by an example demonstrating the exchange of *seed and key*. In this example, a 2-byte *seed* is shown, where the value in the most significant byte is the version number of the implemented standard. The least significant byte is a random value. For the calculation of the corresponding *key*, i.e., the secret, *one's complement* of the seed is used. Standards usually provide an example of the process and some suggestions for an algorithm, e.g. the UDS standard [37] clearly states that the example in its *Security Access* section is just one way of implementing the *secret*.

Sending a key calculated by *one's complement* of the supplied seed, the authentication attempt proved to be successful and unlocked the target. However, in our practical evaluation, we found that this given example in the standard is used in a PCU series unit in-field which contradicts the fact that in the given challenge-response approach the algorithm should be secret. As this algorithm is simple, an attacker can easily check if the given example is implemented. Furthermore, he is also able to easily compromise the PCU by providing the key to the PCU after the seed is sent. Besides using this simple algorithm for the calculation of the *key*, the length is extremely short and not even used to its full capabilities. By using the version number of the standard in the most significant byte, only 256 different *seeds* can be provided by the target as only the least significant byte is changing. This makes it even more vulnerable for exploitation, e.g. by a brute force attack.

To verify the exploitation of this vulnerability we used the test bench in Figure 10 for the selected target and executed all the necessary steps according to the EOL standard. To ensure the functionality of the target, the pyrotechnic charges were simulated by attaching resistors to the corresponding pins on the connector of the target. To enable the EOL functionality CAN traffic was replayed, consisting of only one message including a vehicle speed of zero. This CAN bus only included the target and a CAN transceiver sending the diagnostic messages and playing back the aforementioned vehicle speed message. The exploitation was successfully validated by an oscilloscope displaying the firing impulse. The related Common Vulnerabilities and Exposures (CVE)-ID is listed under CVE-2017-14937 [41]. In addition, a module for the Metasploit Hardware Bridge was developed in cooperation with the security researcher Craig Smith [42]. The module can test for the presence of this vulnerability in a vehicle without the possibility of actually deploy the charges [43].

Lack of Plausibility Checks The observed PCU with the given functionality has access to information from *hard-wired signal sources* and bus messages. However, it does not use all of them to derive if the current state of the vehicle is correct. In fact, it only required one CAN message, containing the vehicle speed, and physical requirements to enable deployment on the test bench used for this work. The physical requirements were connected pyrotechnic charges, in our case the substitution of those by appropriate resistors, and the ignition beeing turned on or the engine running (if installed in a real vehicle). Further attached *hard-wired signals* like seat occupancy were present on the connector of our target but not used.

Combined with the fast and highly automated procedure of the attack, this leads to the possibility of malicious deployment while the engine is running and people reside in the car, e.g. while waiting at a stoplight. This should be seen as a real threat, as the remote exploitation of passenger vehicles, specifically getting access to the CAN bus via remote connections has been demonstrated before [44][45]. Besides, it is conceivable that hackers distribute malicious OBD-Dongles on sales platforms on the Internet. Furthermore, OBD-Dongles sold by manufactures can also give the required access to the internal vehicle network. In a worst-case scenario, this could lead to remote access [46]. Both options could lead to a scaled attack on different types of cars from several manufactures as the vulnerability is part of an international standard.

COUNTERMEASURES In general, software should be designed from the ground up as secure as possible by applying Security by Design during the development process. There are several security principles, which software architects should consider. The following countermeasures are only examples for our observed target based on the mentioned principles.

Selection of Suitable Technologies Documented exploitation of passenger vehicles was achieved in multiple cases by sending or replaying CAN messages on the target's bus. CAN can be seen as a bus system not designed with security issues in mind. Especially the lack of authentication of message sources is one of the main problems of CAN. By using cryptographic authentication methods coupled with integrity protection and a freshness value, this problem can be mitigated. An example of this is an Keyed-Hash Message Authentication Code (HMAC) of the transmitted message, including a freshness value to prevent replay attacks. In addition, techniques such as threat modeling [18][31] can provide valuable information to select specific security measures for implementation. Furthermore, ACL's bidirectional communication capability [47] provides the ability to add an authentication process using a recognized method such as digital certificates. Moreover, the mandatory use of an ACL line on the Airbag-ECU as an additional plausibility check could have prevented exploitation by a single CAN message.

Hard-wired Plausibility Checks The connector of our target was designed to accommodate over 100 *pins*. Although the presence of pins depends on the exact model the target is installed in, this high number is explained by the presence of physical signals being directly connected to the target. Besides the obvious need for *hard-wired connections*, e.g. the connection to the pyrotechnic charges, it was discovered that signals such as *seatbelt status* and *seat occupancy* were connected as well. While this does not exclude the possibility of information being transmitted over a bus system as well, this would allow for plausibility checks based on sources that are not easy to be tampered [48].

Usage of Cryptography When developers want to design a new security feature, they should follow the *Kerckhoff's Principle* [40]. The principle explicitly describes that cryptosystems shouldn't be secure due to hidden details about how the algorithm works (security by obscurity). Moreover, the whole secret should be based on the confidentiality of the used key. The past has already shown multiple times that mechanisms which disregard the mentioned principle are often broken as soon as the algorithm has been reverse-engineered [49]. Our investigation has also confirmed the violation of the mentioned principle by analysing the sent seed. A first approach to fix the existing issue could be an additional security layer which ensures a correct authentication. However, the best solution to the problem is to use generally recommended cryptographic algorithms such as the Advanced Encryption Standard (AES) [50]. The AUTomotive Open System ARchitecture (AUTOSAR) members have already recognized the necessity of the mentioned security goals for future on-board communication. For this reason, they have standardized the SecOC module [51], which includes authentication mechanisms on the level of Protocol Data Units (PDUs). The specification does not define a specific method for creating a Message Authentication Code (MAC), but rather recommends standardized cryptographic algorithms and defines the payload of a secured PDU with a freshness value and an authenticator for protecting against replay attacks and unauthorized manipulation of the message.

Hardening Against Brute-Force Attacks The use of Negative Response Codes (NRCs), such as the *exceeded number of attempts* and *required time delay not expired*, as described in the UDS standard could be used to slow down brute-force attempts. While the former would require a power cycle of the *target* to continue the brute-force attempt, the latter would slow the approach down. To counter the use of scripted power cycling of the target in rapid succession, the target could require a certain time delay after being powered on, before enabling the diagnostic capability. The length of the *seed* and *key* could be increased to harden the target against brute-force attempts. As mentioned before the length of 2 bytes leads to 65536 possible *keys* per *seed*. By using the most significant byte for the version number of the implemented ISO standard, the most significant byte is fixed in every *seed*. Thus, by only changing the values of the least significant byte, only 256 different *seeds* can be supplied by the target after a request, although the seed is 16 bits long. Consequently, this reduces the number of maximum tries to unlock the *Security Access*, as the same *seeds* occur more frequently, which enables a faster iteration over the possible *keys*. Besides this problem, we recommend to increase the key length significantly. In combination with a suitable algorithm such as AES, a key length of 128 bits is recommended [52][53].

Authorization Mechanisms Our own research results regarding the airbag vulnerability have shown again that an unlocked *Security Access* has a high potential for a safety critical impact. Moreover, this type of security mechanism ensures only the authentication. Compared with network systems from the traditional IT, vehicles don't have any mechanisms for authorization and access control. A first approach to implement such features proposed Kim et al. [54] with an approach of an Attribute Based Access Control (ABAC) for the AUTOSAR software architecture based on different attributes of diagnostic CAN messages. Another ABAC approach was published by Berger et al. [55] for integrating this kind of access control in firewall systems. This provides new features such as dynamic access decisions based on environmental conditions such as time or location of the requester. Transferred to automotive systems, it will also be important for future vehicles to implement dynamic and distributed firewalls to address upcoming requirements due to increasing interconnectivity with their infrastructure, other cars or cloud-based functionalities. Furthermore, ABAC could also be useful for enforcing access controls based on different vehicle states. For instance, in one of our tested vehicles, the gateway did not check for the appropriate conditions, before performing certain diagnostic functions. As a consequence of this, diagnostic functions are available during vehicle conditions in which there would be malicious applications for them. This includes entering a diagnostic session specifically for safety systems, e.g. airbags, at highway speeds. Thus, it is probable that even more implementation flaws exist in our target which are yet to be discovered.

CONCLUSION AND FUTURE WORK

In this work we have developed an approach to reuse the results of a threat analysis for automotive security testing. A structured methodology for deriving test cases using attack trees was presented. In addition, the presented approach shows how exactly the results of a safety analysis can be reused to identify threats that can compromise vehicle safety. To identify this particular type of threat, we have shown how our recently published approach can be used. Furthermore, we have provided a mapping between the nomenclature of our threat identification approach and STRIDE for an easier integration into existing threat modeling methods. Finally, we demonstrated the applicability of our method by an experimental evaluation with an Airbag-ECU. In particular, the preparation and execution of the penetration tests for the Airbag-ECU demonstrated the applicability of the proposed methodology steps. As a result of this penetration test, a vulnerability was discovered that allows unintentional detonation of airbags.

In the next steps, we want to further formalize our approach. This could help us to create test cases with a detailed description for the concrete selection of attack vectors. Such a detailed test case description can further be used to develop specific testing tools. In general, we want to offer a computer-aided version of our approach to reduce the total testing effort. Furthermore, we want to analyse a possible reuse of functional requirements as further input for our approach.

APPENDIX

For an easier integration in existent approaches we want to provide a mapping between SGM guide-words and the CIA(A) triad in Table 4. This further allows us to provide a mapping between SGM and STRIDE. We provide this transformation ability between SGM and STRIDE due to the fact that STRIDE nomenclature is well-known in security and commonly used. As one example for the mapping, we utilize the threat *Unintended airbag deployment can be triggered by triggering of (diagnostic) function airbag deployment.* that was built with the guide-word *triggering of (diagnostic) function*. For this guide-word, we propose to select *spoofing* from Table 4 as pedant for STRIDE. With this, we can transform the threat description to *Spoofing of (diagnostic) function* presenting the STRIDE nomenclature. We do this for the other two threats as well, leading to an attack tree in STRIDE nomenclature shown in Figure 12.

Table 4: Mapping from SGM to STRIDE nomenclature, with the CIA(A) triad as intermediate mapping step

SGM	CIA(A)	STRIDE
triggering	authenticity, integrity	spoofing
insertion	integrity	tampering
manipulation	integrity	tampering
disconnection	availability	denial-of-service
delay	availability	denial-of-service
deletion	availability	denial-of-service
stopping	availability	denial-of-service
reset	availability	denial-of-service

ACKNOWLEDGEMENTS

This work has been developed in the projects SAFE ME ASAP (reference number: 03FH011IX5) and AUTO-

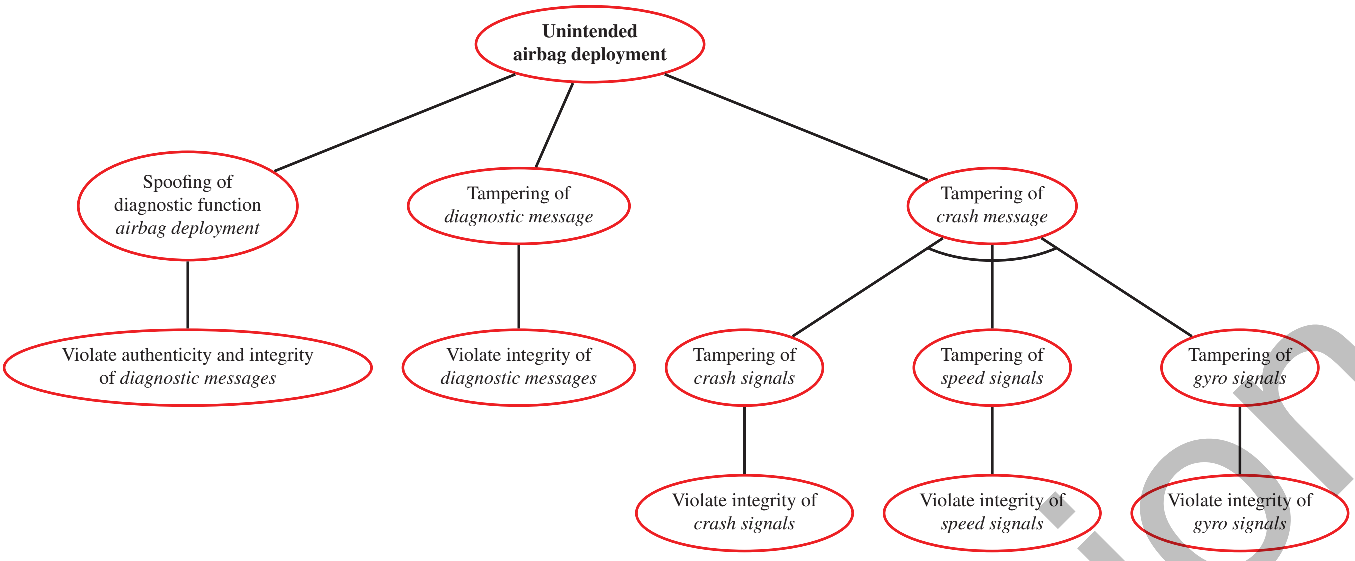
Unintended airbag deployment
Spoofing of diagnostic function *airbag deployment*
Tampering of *diagnostic message*
Tampering of *crash message*
Violate authenticity and integrity of *diagnostic messages*
Violate integrity of *diagnostic messages*
Tampering of *crash signals*
Tampering of *speed signals*
Tampering of *gyro signals*
Violate integrity of *crash signals*
Violate integrity of *speed signals*
Violate integrity of *gyro signals*

Figure 12: Attack tree for the threat *Unintended airbag deployment* after the transformation to the STRIDE nomenclature by using the mapping in Table 4

SIMA (reference number: 13FH006IX6) which are partly funded by the German ministry of education and research (BMBF) within the research programme ICT 2020.

References

[1] R. N. Charette, "This Car Runs on Code," 2009, accessed 12.02.2016. [Online]. Available: http://spectrum.ieee.org/transportation/systems/this-car-runs-on-code

[2] C. Miller and C. Valasek, "A survey of remote automotive attack surfaces," *Black Hat USA*, vol. 2014, 2014.

[3] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno *et al.*, "Comprehensive Experimental Analyses of Automotive Attack Surfaces," in *USENIX Security Symposium*, 2011.

[4] C. Valasek and C. Miller, "CAN Message Injection: OG Dynamite Edition," last checked on 05.04.2017. [Online]. Available: http://illmatics.com/can%20message%20injection.pdf

[5] A. Greenberg, "Tesla Responds to Chinese Hack With a Major Security Upgrade," last checked on 23.03.2017. [Online]. Available: https://www.wired.com/2016/09/tesla-responds-chinese-hack-major-security-upgrade/

[6] H. H. Thompson, "Application penetration testing," *IEEE Security and Privacy Magazine*, vol. 3, no. 1, pp. 66–69, 2005.

[7] M. Felderer, M. Büchler, M. Johns, A. D. Brucker, R. Breu, and A. Pretschner, "Security testing: A survey," in *Advances in Computers*. Elsevier, 2016, vol. 101, pp. 1–51.

[8] ISECOM, "Open source security testing methodology manual," Internet. [Online]. Available: http://www.isecom.org/research/

[9] NIST, "Technical guide to information security testing and assessment," Internet. [Online]. Available: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-115.pdf

[10] OISSG, "Information systems security assessment framework," Internet. [Online]. Available: http://cuchillac.net/archivos/pre_seguridad_pymes/2_hakeo_etico/lects/metodologia_oissg.pdf

[11] F. Gontharet, "ISSAF – Methodology Analysis and Critical Evaluation)," 2015. [Online]. Available: https://wr0ng.name/other/REPORT_PenetrationTesting_Methodology.pdf

[12] PTES, "Penetration testing execution standard," Internet. [Online]. Available: http://www.pentest-standard.org

[13] OWASP, "Open web application security project - testing guide v4," Internet. [Online]. Available: https://www.owasp.org/images/1/19/OTGv4.pdf

[14] SAE, "Cybersecurity Guidebook for Cyber-Physical Vehicle Systems," 01.2016, accessed 12.04.2016. [Online]. Available: http://standards.sae.org/wip/j3061/

[15] S. Plósz, C. Schmittner, and P. Varga, “Combining Safety and Security Analysis for Industrial Collaborative Automation Systems,” in *International Conference on Computer Safety, Reliability, and Security*, 2017, pp. 187–198.

[16] S. P. Kadhirvelan and A. Söderberg-Rivkin, “Threat Modelling and Risk Assessment Within Vehicular Systems,” Masterthesis, CHALMERS UNIVERSITY OF TECHNOLOGY, Schweden, 2014. [Online]. Available: http://publications.lib.chalmers.se/records/fulltext/202917/202917.pdf

[17] A. Ruddle, D. Ward, B. Weyl, S. Idrees, Y. Roudier, M. Friedewald, T. Leimbach, A. Fuchs, S. Gürgens, O. Henniger *et al.*, “Deliverable d2. 3: Security requirements for automotive on-board networks based on dark-side scenarios,” *tech. rep., EVITA*, 2009.

[18] “The STRIDE Threat Model,” 2005. [Online]. Available: https://msdn.microsoft.com/en-us/library/ee823878(v=cs.20).aspx

[19] A. Shostack, *Threat modeling: Designing for security*. John Wiley & Sons, 2014.

[20] O. Henniger, L. Apvrille, A. Fuchs, Y. Roudier, A. Ruddle, and B. Weyl, “Security requirements for automotive on-board networks,” in *Intelligent Transport Systems Telecommunications,(ITST), 2009 9th International Conference on*. IEEE, 2009, pp. 641–646.

[21] M. Nancy, S. Forrest, V. Krishnamurthy, and V. Ole, “A Hybrid Threat Modeling Method: TECHNICAL NOTE,” Internet. [Online]. Available: https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=516617

[22] T. Denning, B. Friedman, and T. Kohno, “The Security Cards: A Security Threat Brainstorming Toolkit,” *Univ. of Washington, http://securitycards.cs. washington. edu*, 2013. [Online]. Available: http://securitycards.cs.washington.edu/index.html

[23] G. Macher, E. Armengaud, E. Brenner, and C. Kreiner, “Threat and Risk Assessment Methodologies in the Automotive Domain,” *Procedia Computer Science*, vol. 83, pp. 1288–1294, 2016.

[24] G. Macher, H. Sporer, R. Berlach, E. Armengaud, and C. Kreiner, “SAHARA: a security-aware hazard and risk analysis method,” in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2015, pp. 621–624.

[25] C. Schmittner, Z. Ma, and P. Smith, “FMVEA for Safety and Security Analysis of Intelligent and Cooperative Vehicles,” in *Computer safety, reliability, and security*, ser. LNCS Sublibrary: SL 2 - Programming and Software Engineering, A. Bondavalli, A. Ceccarelli, and F. Ortmeier, Eds. Heidelberg: Springer, 2014, vol. 8696, pp. 282–288.

[26] C. Schmittner, Z. Ma, E. Schoitsch, and T. Gruber, “A Case Study of FMVEA and as Safety and Security Co-Analysis Method for Automotive Cyber-physical Systems,” in *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*, 2015, pp. 69–80.

[27] R. Fredriksen, M. Kristiansen, B. A. Gran, K. Stølen, T. A. Opperud, and T. Dimitrakos, “The CORAS framework for a model-based risk management process,” in *International Conference on Computer Safety, Reliability, and Security*, 2002, pp. 94–105.

[28] S. Kriaa, L. Pietre-Cambacedes, M. Bouissou, and Y. Halgand, “A survey of approaches combining safety and security for industrial control systems,” *Reliability Engineering & System Safety*, vol. 139, pp. 156–178, 2015.

[29] W. Young and N. G. Leveson, “An integrated approach to safety and security based on systems theory,” *Communications of the ACM*, vol. 57, no. 2, pp. 31–35, 2014.

[30] I. Friedberg, K. McLaughlin, P. Smith, D. Laverty, and S. Sezer, “STPA-SafeSec: Safety and security analysis for cyber-physical systems,” *Journal of Information Security and Applications*, vol. 34, pp. 183–196, 2017.

[31] J. Dürrwang, K. Beckers, and R. Kriesten, “A Lightweight Threat Analysis Approach Intertwining Safety and Security for the Automotive Domain,” in *International Conference on Computer Safety, Reliability, and Security*, 2017, pp. 305–319. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-66266-4_20

[32] R. Winther, O.-A. Johnsen, and B. A. Gran, “Security assessments of safety critical systems using HAZOPs,” in *Computer Safety, Reliability and Security*. Springer, 2001, pp. 14–24.

[33] R. Winther, “Qualitative and Quantitative Analysis of Security in Safety and Reliability Critical Systems,” in *Probabilistic Safety Assessment and Management*, 2004, pp. 2345–2351.

[34] J. Dürrwang, “Evaluation Security Guideword Experiment,” Internet, accessed 13.03.2017. [Online]. Available: http://www.home.hs-karlsruhe.de/~duju0001/Evaluation_SGM/

[35] GitHub, “Caringcaribou,” Internet, accessed 13.02.2018. [Online]. Available: https://github.com/CaringCaribou/caringcaribou

[36] M. Ring, J. Dürrwang, F. Sommer, and R. Kriesten, "Survey on vehicular attacks-building a vulnerability database," in *Vehicular Electronics and Safety (ICVES), 2015 IEEE International Conference on*. IEEE, 2015, pp. 208–212.

[37] ISO, "14229 Unified diagnostic services (UDS)," 2012.

[38] ——, "ISO 26021-2:2008 Road vehicles – End-of-life activation of on-board pyrotechnic devices," 2009.

[39] M. Ring, T. Rensen, and R. Kriesten, "Evaluation of Vehicle Diagnostics Security: Implementation of a Reproducible Security Access," *Secureware*, vol. 2014, 2014.

[40] A. Kerckhoffs, *La cryptographie militaire, ou, Des chiffres usités en temps de guerre: Avec un nouveau procédé de déchiffrement applicable aux systèmes à double clef*. Librairie militaire de L. Baudoin, 1883.

[41] "CVE-2017-14937," 2017. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-14937

[42] Rapid7, "Metasploit Wrapup: Airbag Authentication," 2017. [Online]. Available: https://blog.rapid7.com/2017/12/22/metasploit-wrapup-21/

[43] ——, "CVE-2017-14937 Check For and Prep the Pyrotechnic Devices (Airbags, Battery Clamps, etc.)," 2017. [Online]. Available: https://www.rapid7.com/db/modules/post/hardware/automotive/pdt

[44] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, 2015.

[45] Keen Security Lab, "Experimental Security Assessment of BMW Cars: A Summary Report," 2018. [Online]. Available: https://keenlab.tencent.com/en/Experimental_Security_Assessment_of_BMW_Cars_by_KeenLab.pdf

[46] A. Kovelman, "A Remote Attack on the Bosch Drivelog Connector Dongle - Argus Cyber Security," 2017. [Online]. Available: https://argus-sec.com/remote-attack-bosch-drivelog-connector-dongle/

[47] ISO, "Road vehicles – End-of-life activation of on-board pyrotechnic devices: Part 4: Additional communication line with bidirectional communication," 05.2009. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso:26021:-4:ed-1:v1:en

[48] J. Dürrwang, M. Rumez, J. Braun, and R. Kriesten, "Security Hardening with Plausibility Checks for Automotive ECUs," in *VEHICULAR 2017*, 2017, vol. 6, pp. 38–41. [Online]. Available: http://www.thinkmind.org/download.php?articleid=vehicular_2017_2_40_30053

[49] C. Paar and J. Pelzl, *Understanding cryptography: A textbook for students and practitioners*. Heidelberg and New York: Springer, ©2010.

[50] A. K. Lenstra and E. R. Verheul, "Selecting cryptographic key sizes," *Journal of Cryptology*, vol. 14, no. 4, pp. 255–293, 2001.

[51] AUTOSAR, "Specification of Secure Onboard Communication: AUTOSAR CP Release 4.3.1," 2017. [Online]. Available: https://www.autosar.org/fileadmin/user_upload/standards/classic/4-3/AUTOSAR_SWS_SecureOnboardCommunication.pdf

[52] M. Margraf, "Kryptographische Verfahren: Empfehlungen und Schlüssellangen," *Technische Richtlinie TR-02102, Bundesamt fur Sicherheit in der Informationstechnik*, 2008.

[53] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "Recommendation for key management part 1: General (revision 3)," *NIST special publication*, vol. 800, no. 57, pp. 1–147, 2012.

[54] D.-K. Kim, E. Song, and H. Yu, "Introducing attribute-based access control to autosar. no. 2016-01-0069," *SAE Technical Paper*, 2016.

[55] S. Berger, A. Vensmer, and S. Kiesel, "An abac-based policy framework for dynamic firewalling," in *International Conference on Systems and Network Communications (ICSNC 2012), 2012*, 2012, pp. 118–123.