Towards a security-by-design approach enabling automated validation in automotive architectures

Marco Ebster, Boris Brankovic*, Katharina Polanec, Christoph Binder, Christian Neureiter

Abstract Securing automotive architectures against cyber-attacks is a challenging task, especially if function and safety have to be considered and the overhead of the security engineering process has to be minimal. Security-by-design will become a necessary property of automotive architectures in the future, because of rising vehicle interconnection and the associated risks. ThreatGet already provides an approach for automated threat analysis and risk assessment in this field. Nevertheless, there is still a lack of automation in the area of cybersecurity risk remediation and subsequent result validation or verification. Therefore, this paper combines the security pattern engineering process with ThreatGet and the Automotive Reference Architecture Model (ARAM) Framework to provide a multi-layered top-down approach to consider the parallel development of the function, safety, and security in automotive architectures according to ISO-21434.

1 Introduction

In the last decades the automotive industry has become one of the world's most significant economic sectors according to [16] by producing nearly 95 billion units (including passenger cars and commercial cars) and with more than 80 millions vehicles sold yearly. Another key aspect reflected in [1] is the rapidly increasing number of Electric Vehicles (EVs) and the impact of EVs on the automotive market as those are seen as replacement of fossil-energy-driven cars crucial to mitigate air pollution, oil dependency and green house gas emissions. However, not only the automotive market is facing an enormous growth, but also the amount of hardware and software components within cars has dramatically increased - nowadays more than

Marco Ebster, Boris Brankovic, Katharina Polanec, Christoph Binder, Christian Neureiter
Josef Ressel Centre for Dependable System-of-Systems Engineering - Salzburg University of Applied Sciences, Urstein Süd 1, A–5412 Puch/Salzburg, Austria, e-mail: {mebster.its-m2020, boris.brankovic, katharina.polanec, christoph.binder, christian.neureiter}@fh-salzburg.ac.at

hundred Electronic Control Units (ECUs) and hundred millions lines of code are part of modern vehicles. For example the 2017 Ford F150 has already exceeded 150 millions lines of code, as the trend towards autonomous driving requires new features to be implemented and thus present and future vehicles can be defined as Complex Autonomous Vehicles (CAVs) [16, 19, 11]. Furthermore, the authors in [2] have investigated possible scenarios in relation to CAVs like *Vehicle Platooning*, where multiple self-driving vehicles are travelling together at high-speed for a specific distance. Therefore, communication technologies like Vehicle-to-Vehicle (V2V) will be used in future *Smart Cities* to enable an effective cooperation between CAVs and to avoid potential crashes. Moreover, the work in [22] underpins even more the importance of real-time communication in traffic flows with the development of Vehicle-to-Infrastructure (V2I) communication techniques enabling for instance a seemless interaction between CAVs and traffic light controller systems. Furthermore, the interaction of EVs with Smart Grids (SGs) plays also a significant role concerning the communication infrastructure in the context of *Smart Cities*. Consequently, the authors in [17] have researched on EVs serving as independent energy source for the SG and the concept of Vehicle-to-Grid (V2G). All the mentioned communication protocols viz. V2V, V2I and V2G can be summarized by the term Vehicle-to-Everything (V2X). It contemplates the current communication landscape of modern vehicles and shows the diversity of interactions with other systems, also referred as V2X communication environment.

Thus, vehicles interacting with their vicinity can not be seen as isolated systems anymore, but as a System-of-Systems (SoS), which especially increases the dependability and with that the attack surface of these systems. With reference to Neureiter [18] dependability intends to minimize risks emerging from systems which lead to harms of systems in the same context. Accordingly, the author inventend a set of dependability requirements, assuring safety and security associated harms like *malicious attacks* from the outside can already be considered in the design-process phase of systems. Aditionally, the authors in [23] and [12] examined various cybersecurity attacks in relation to CAVs and support the significance of Security-by-Design approaches, as it is essential to analyze potential attack vectors and to eliminate emerging risks prior to the development of systems. Hence, the international standard for cybersecurity ISO-21434 [10] suggests the development of an preliminary architecture for the purpose of the identification of an specific item, together with its operational environment and their relationships in the context of cybersecurity. With the definition of domain-specific reference architectures like the ARAM Framework [5] it is possible to create architectures in the automotive domain by utilizing Model Based Systems Engineering (MBSE) concepts. This framework should serve as base for the preliminary architecture definition and for the investigation on possible countermeasures with respect to cybersecurity attacks on CAVs. Therefore, this paper proposes the definition of design-countermeasures in the form of Design-Patterns based on the ISO-21434 to mitigate cyberattacks on certain components of CAVs and thus to recognize and eliminate risks in the design-phase of systems.

To address the mentioned aspects, this paper is structured as follows: Section 2 outlines the related work about cybersecurity and security-by-design approaches. Further, in section 3 the research approach is described in more detail. Section 4 and 5 respectively explain the implementation and application of design countermeasures. Finally, in section 6, the findings of the paper are outlined and a conclusion is given.

2 Related Work

2.1 (Cyber-)Security-By-Design

According to the studies of Haberfellner et al. [9], systems with a high variety of interconnections and a large number of interrelated elements can be seen as complex systems. In general CAVs can be classified as such a complex system and as a specific form of Cyber-Physical Systems (CPS) and Internet of Things (IoT) system, due to their characteristics. Typically, CPS consist of a distributed set of sensor nodes forming a feedback loop enabling for example full autonomy, or at least semi-autonomous functions. The close connection between CAVs and CPS was considered in more detail in [3], where the authors have investigated on security issues related to CPS and point out the importance of security-by-design approaches. Moreover, the results have shown that it is not sufficient to consider only the attack surface on CPS, as certain aspects like the generalization of attacks usually ignores security-by-design fundamentals e.g., the role of Roots of Trust (ROT). Furthermore, in terms of CAVs the main security goal is the resilience of safety features of autonomous functions, as human lives can be harmed by any compromise of the functionalities provided by CAVs. Thus, it is important to identify the entire system life-cycle of a CPS, including for instance all steps and people involved in the security thought of the system. This can be done by the utilization of a hollistic approach like Systems Engineering (SE), where the key concepts of designing security for complex systems can be described within a system model. Following the results in [3] two potential approaches for designing security exist to the time of their research viz. the development of security policies guided by the Confidentiality, Integrity and Availability requirements, also known as CIA triad, or the creation of the STRIDE threat model invented by Microsoft. Additionally, the previously mentioned authors itself propose as well a security-by-design approach for autonomous vehicles by combining key principles from the standards ISO-26262 and SAE-J3061 [13]. To time another state-of-the-art standard was introduced: the ISO-21434 [10]. It provides inter alia mitigating strategies throughout the automotive field and aims to deliver safe and secure CAVs.

2.2 Automotive Systems Engineering

As systems are becoming more and more complex due to the rising interconnection with other systems and the interplay of different teams during the system development phase, the need for interdisciplinary approaches utilizing proven concepts to overcome the growing complexity in systems, is essential. Therefore, the discipline of SE is used as established approach for the modeling of complete system architectures, in order to create new applications and to satisfy the needs of stakeholders [8]. Various SE frameworks exist that can be used in different domains to perform MBSE tasks, aiming to define one central element i.e. a system model that captures all required information and provides consistency through all stages of development [24, 25]. The ARAM Framework for instance, is a result of the evaluation of different automotive frameworks as stated in [5] and allows to perform model-based and domain-specific SE within the automotive domain. Furthermore, with the three-dimensional structure it is possible to model entire CAV architectures from different perspectives and on diverse layers of abstractions. This enables to model multiple aspects of a specific System of Interest (SOI) from the bare definition of requirements to the realization with suitable components. Moreover, in terms of cybersecurity, the ARAM Framework can be supplemented by the threat modeling based approach *ThreatGet*, developed by the Austrian Institute of Technology (AIT) [20]. Thus, with *ThreatGet* it is feasible to perform an automated risk identification and to discover potential attack vectors concerning CAVs.

2.3 Security Patterns as a derivative of Design Patterns

Design patterns provide solutions that solve explicit problems by using proven concepts. They aim to avoid time and knowledge-consuming examination of already solved problems. Design patterns generalize a solution so that it can be reused for well-known problems in similar fields. Security patterns, a derivative of Design patterns, provide concepts for security designs and best practices from an existing body of knowledge to solve security problems in related scenarios. ”Security patterns are represented as textual templates or combined with Unified Modeling Language (UML) models, in a hierarchically layered architecture or in a searchable pattern library.”[15] Security pattern engineering begins with the analysis of the system architecture and the determination of regarding security vulnerabilities and threats. Established security analysis methods and techniques, to reveal them, are attack surface analysis, attack trees, and threat modeling (ThreatGet). Subsequently, appropriate mitigation controls for the identified vulnerabilities and threats must be applied. Finally, the result is verified and validated against context-related standards. If the security pattern is approved, it can be instantiated and applied to the existing system architecture design. A security pattern is a generalization of a security design, therefore it is crucial to take into consideration the context of an application. The outcome of the security pattern engineering process is a secure system architecture [15]. According to ISO-21434 and Threat Analysis and Risk Assessment (TARA) methods [10], security goals can be specified from the system architecture after the security pattern application and subsequently security requirements can be derived from it. The combination of the security pattern engineering process, the definition of security goals and requirements lead to an ultimately cyber-security concept [10].

3 Approach

3.1 Agile Design Science Research Methodology

As mentioned before, the main goal of this work is to define design countermeasures for the mitigation of cyberattacks in the design-phase of CAVs. However, the definition of quantifiable requirements can be difficult to elicit in such an agile development scenario, because of the uncertain specification of the final result. Therefore, the ideas of the Agile Design Science Research Methodology (ADSRM) proposed in [4] were adopted, as this approach fosters the constant improvement of both, the problem- and solution space in an dynamically changing application scenario. In each iteration of the methodology, defined artifacts, as well as requirements are evolving, where each iteration itself provides a solution of the intended artifacts. Thus, a specific process is introduced by this very methodology, containing various process steps for the development of the artifacts and offering several entry points that can be used to initiate the iterative process model. With the usage of this method, agile software development is supported and for the certain case in this work the iteration cycle is joined by the so-called *design and development-centered* initiation. Following the definition in [4] the first input of the ADSRM process is an appropriate case study, essential for the requirements engineering process and for the development of the main artifacts. Therefore, this paper uses a case study regarding an Over The Air (OTA) update and the resulting effect on the Braking System of an CAV. Thus, the main artifact concerns the development of a security pattern based on the ISO-21434, which can be applied to models created with the ARAM Framework and in combination with the threat modeling tool ThreatGet. Consequently, the case study is explained more particularly in the following.

Case Study - Referring to the previously mentioned research methodology a suitable case-study is demanded by the ADSRM process to develop the main artifacts. Based on a scientific cooperation with the International Council on Systems Engineering (INCOSE) automotive working group, the case about a Braking System of an CAV was chosen, as this study has an particular impact on safety-related CPS in the automotive area. The case was first described in [21] and explains in detail the Braking System of an autonomous vehicle, with its functionality and subcomponents needed to perform braking in certain operational moments.

Further, this study can be complemented with a variety of scenarios occuring in future Smart Cities stated in [2] e.g., *Vehicle Platooning*. Moreover, to address security related aspects, the selected study has been supplemented by the integration of an OTA update scenario, as it especially contemplates possible impacts on the CAV Braking System while performing an update. This can be for instance, a general software update, or the update of a specific component firmware within the vehicle, via an external server. Findings in [14] have shown, that an OTA update triggers new security challenges, as with such an update vehicle security can be compromised.

4 Implementation

The previously mentioned security pattern engineering process provides a high-level concept of how to construct a security pattern for automotive architectures according to ISO-21434 [6]. The basis for the security design process is the previously mentioned case study about the CAV Braking System, which was modeled with the ARAM Framework. The security pattern is based on a snapshot of the system from the point of view of ARAMs Information Layer. This approach aims to incorporate the security design process into architectural development as early as possible. This enables top-down refinement of the security pattern, from layer to layer. Analogously, conclusions can already be drawn about the quality of the modeled architecture in terms of security on each layer and an outlook on required components can already be given for the underlying layers. The output of the security pattern application can be validated and verified against the ISO-21434 on every layer. Constructing the pattern is shown in a five-step process:

Step 1: Identify the problem space and scope: The following security pattern describes controls required to ensure the fail-safety of a CAVs braking system during an OTA software update. OTA connections between in-vehicle and external networks are utilized for remote maintenance, remote control, or software updates. The scope of this procedure is to address the security threats that relate to:

- Communication between vehicle and Original Equipment Manufacturer (OEM) backend.
- In-vehicle communication and system reliability.

Step 2: Prepare and research: In the previous step, an initial view of the problem space and scope was formed. This step is intended to form the basis for the threat modeling and find initial controls required to mitigate those threats. Before the development of ThreatGet, that meant collecting available research for reference or building out new research and analysis [6]. The following steps focus on the main threats and recommended controls for OTA software updates, which were defined in [14] and are displayed below.

Threats, Risks and/or Vulnerabilities	Anticipated Security Controls
• Read Updates: Attackers access/read contents of software updates, which may lead to loss of intellectual property.	• An encrypted channel is used between the vehicle and the OEM system.
• Deny Updates: Attackers prevent a vehicle from fixing software problems by denying access to software updates.	• Enhanced cryptography and image stenography techniques for software updates.
• Deny Functionality: Attackers cause a vehicle to fail to function.	• Verify the authenticity of the software update source
• Control: Attackers cause a vehicle to install software of their choosing.	• Use automotive software update frameworks (Uptane).
	• Log update status and events.

A list of assets affected by the problem statement was also created during research.

Asset Title	Asset Description
• Software Update Server (External)	• OEM server contains the software update repository.
• CAV Central Gateway (Internal)	• Orchestration between Server/Client infrastructure. Requests and retrieves software updates.
• Software Update Client (Internal)	• CAV component, deploys and acknowledges software updates.

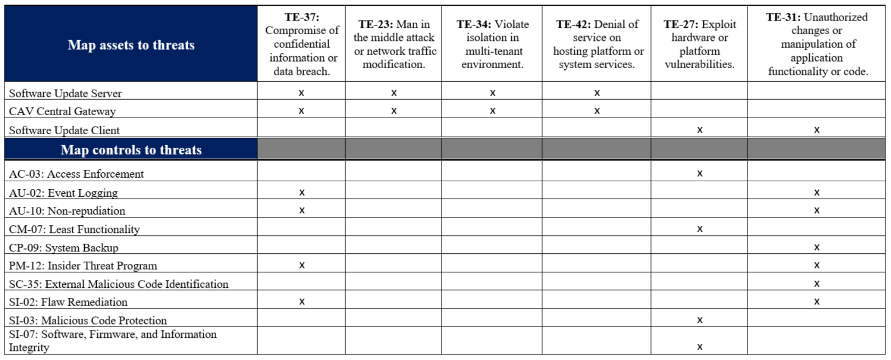
Step 3: Threat modeling: This step provides a list of threats within the problem statement [6]. The OTA software update process modeled with the ARAM Framework is the target for threat modeling using the threat modeling tool ThreatGet based on the automotive ruleset. The identified threats were categorized against the threat events taxonomy. For each threat the potential threat events were characterized, the relevance of the event assessed and the threat sources determined that could initiate the events [6]. The threat modeling produced the following result. As outlined in the previous step this is not the overall list of threats identified by ThreatGet. It concentrates on the main risks of OTA software updates.

Threat Event (ID / Title)	Threat Description and Characteristics
• TE-37: Compromise of confidential information or data breach. TE-23: Man in the middle attack or network traffic modification.	• Software update content maybe is intercepted or stolen. One way to read this data is to stage an eavesdropping attack.
• TE-34: Violate isolation in multi-tenant environment.	• Blocking/Slowing-down software update retrievals lead to outdated software or exploitation of known vulnerabilities.
• TE-42: Denial of service on hosting platform or system services.	• Continuous sending of a certain software update may lead to update prevention, freezing, or a crash of a vehicle component.
• TE-27: Exploit hardware or platform vulnerabilities.	• Sending outdated or incompatible software with known vulnerabilities may lead to the vehicle failing to function.
• TE-31: Unauthorized changes or manipulation of application functionality or code.	• Overwriting the software update with malicious software may lead to unanticipated behavior.

Step 4: Define and map security control objectives: In this step, the complete research done up to this point is used to determine appropriate controls to mitigate the threats. It is important to note that this step does not automatically generate an answer to what security controls are required within a security pattern. It ensures that all different angles of the problem space are considered and that completeness is built into the rationale as to why those controls are needed [6]. The National Institute of Standards and Technology (NIST) Risk Management Framework is used as a taxonomy for security controls. First, threats are mapped to assets. In the upper part of Figure 1, each threat is stepped through and the affected assets are identified. Second, controls are mapped to threats. In the lower part of Figure 1, for each threat, the mitigating controls are identified. To avoid repetitive work and to not go beyond the scope of this paper only the asset *Software Update Client* is considered from this point [6].

Step 5: Assemble the security pattern: After the mapping of threats to assets and threats to controls, the relationship between assets and controls is built out. In this step, the two different mappings are reversely traversed from the control objectives back to the asset. This results in a list of controls to mitigate the identified threats for the asset, *Software Update Client* [6].

Threat Event	**Security Controls Objectives**
• TE-27: Exploit hardware or platform vulnerabilities.	• AC-03: Access Enforcement, CM-07: Least Functionality, SI-03: Malicious Code Protection, SI-07: Software, Firmware, and Information Integrity
• TE-31: Unauthorized changes or manipulation of application functionality or code.	• AU-02: Event Logging, AU-10: Non-repudiation, PM-12: Insider Threat Program, SC-35: External Malicious Code Identification, SI-02: Flaw Remediation, CP-09: System Backup



Map assets to threats	**TE-37:** Compromise of confidential information or data breach.	**TE-23:** Man in the middle attack or network traffic modification.	**TE-34:** Violate isolation in multi-tenant environment.	**TE-42:** Denial of service on hosting platform or system services.	**TE-27:** Exploit hardware or platform vulnerabilities.	**TE-31:** Unauthorized changes or manipulation of application functionality or code.
Software Update Server	x	x	x	x		
CAV Central Gateway	x	x	x	x		
Software Update Client					x	x
Map controls to threats						
AC-03: Access Enforcement					x	
AU-02: Event Logging	x					x
AU-10: Non-repudiation	x					x
CM-07: Least Functionality					x	
CP-09: System Backup						x
PM-12: Insider Threat Program	x					x
SC-35: External Malicious Code Identification						x
SI-02: Flaw Remediation	x					x
SI-03: Malicious Code Protection					x	
SI-07: Software, Firmware, and Information Integrity					x	

Fig. 1: Threats mapped to assets / Controls mapped to threats

Control families: AC - Access Control, AU - Audit and Accountability, CM - Configuration Management, CP - Contingency Planing, PM - Program management, SC - System and Communication Protection, SI - System and Information Integrity

5 Application

As explained above, the goal is to further automate the security-by-design of automotive architectures according to ISO-21434. The first major step to achieve was the automation of threat modeling. This has already been addressed with ThreatGet. The use case considered in this paper is the OTA software update process. Figure 2 contemplates an example, modeled with the ARAM Framework and implemented as Domain Specific Language (DSL) for the modeling software Enterprise Architect (EA). The bottom diagram shows the modeling of the ARAM Information Layer, which is the starting point for the application of ThreatGet. The ARAM layer is mapped to a proprietary ThreatGet diagram, see the top diagram of Figure 2. After that, the automated threat analysis and risk assessment is performed. Where ThreatGet ends, the next step towards further automation of security-by-design begins. Accordingly, the NIST Risk Management Framework provides the needed control objectives to mitigate the identified threats. At the end of the previous section, it is shown, how security patterns can be assembled. However, keep in mind, that each control objective described under NIST SP 800-53 [7] is written as a generic statement for a broad set of assets [6]. This is an important property for developing a general security pattern, but before the controls can be applied, they must first be written in the context of the asset.

Asset Title	Asset Description
• AC-03: Access Enforcement	• Enforce discretionary access restrictions for all devices, and services, which try to interact with the client.
• CM-07: Least Functionality	• Clients work only with minimal services and toolsets required for their usage.
• SI-03: Malicious Code Protection	• Clients have endpoint protection enabled for anti-malware and intrusion prevention.
• SI-07: Software, Firmware, and Information Integrity	• Ensure system integrity for CAV core functionality is maintained.
• AU-02: Event Logging Malicious Code Protection	• Clients that generate security event records are aggregated into a centralized security event monitoring.
• AU-10: Non-repudiation Event Logging Malicious Code Protection	• Firmware and devices are digitally signed from a trusted source (OEM).
• PM-12: Insider Threat Program Event Logging Malicious Code Protection	• Regularly monitor clients at regular intervals to ensure compliance with workflows and the level of access permissions.
• SC-35: External Malicious Code Identification	• Periodically scan for potentially malicious code that may be introduced to the external firmware database of the OEM.
• SI-02: Flaw Remediation	• Any firmware detected with security threats is not permitted to deploy on the client.
• CP-09: System Backup	• Using an A/B partition method will provide a failsafe for the update process.

Subsequently, the security pattern is modeled as a Design Pattern in EA using the previously assembled asset-centric controls. Thereafter, the security pattern is added to the ARAM Toolbox. There it is part of the ISO-21434 extension, which provides templates for common security problems in automotive architectures. Using the toolbox, security patterns can be applied to ARAM diagrams via drag and drop. The result of the application to the OTA software update architecture, which was modeled on the ARAM Information Layer, can be seen in the right corner of the bottom diagram in Figure 2. The security pattern consists of a list of asset-centric controls that must be met to mitigate the risks identified by threat modeling. It should be noted that the security pattern is currently modeled for only one asset, the Software Update Client. However, it is already a preview of the result.

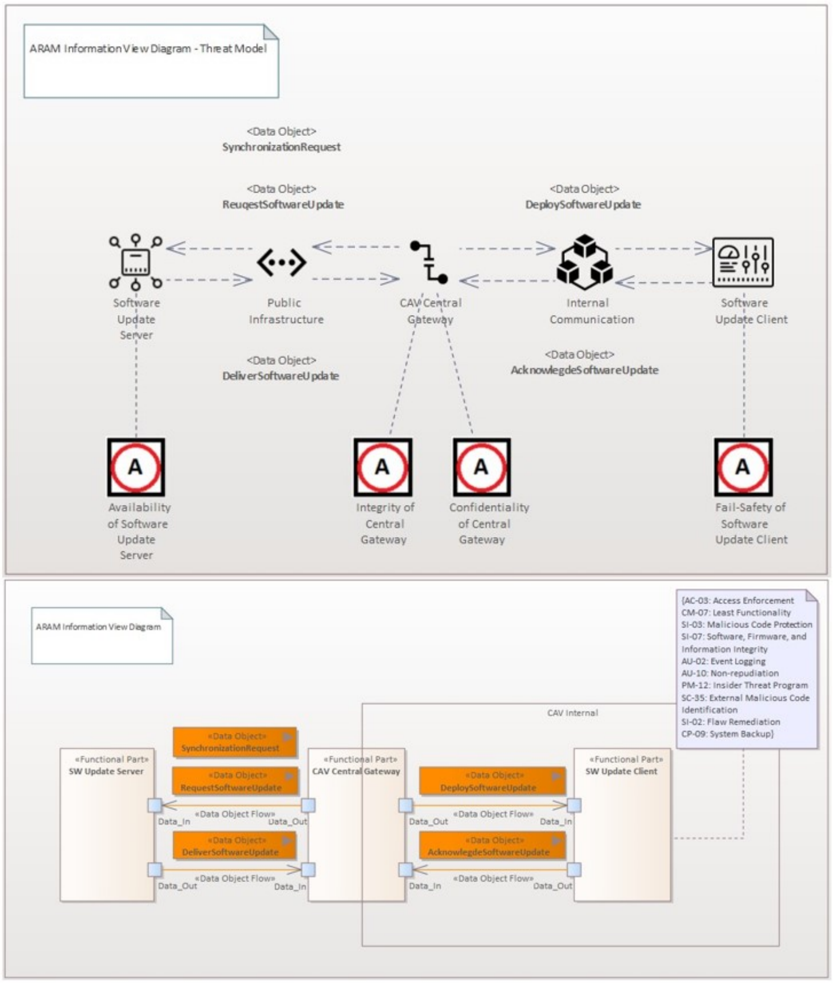

ARAM InformationView Diagram - Threat Model
<Data Object> SynchronizationRequest
<Data Object> ReuqestSoftwareUpdate
<Data Object> DeploySoftwareUpdate
Software Update Server
Public Infrastructure
CAV Central Gateway
Internal Communication
Software Update Client
<Data Object> DeliverSoftwareUpdate
<Data Object> AcknowlegdeSoftwareUpdate
A
Availability of Software Update Server
Integrity of Central Gateway
Confidentiality of Central Gateway
Fail-Safety of Software Update Client
ARAM Information View Diagram
{AC-03: Access Enforcement
CM-07: Least Functionality
SI-03: Malicious Code Protection
SI-07: Software, Firmware, and Information Integrity
AU-02: Event Logging
AU-10: Non-repudiation
PM-12: Insider Threat Program
SC-35: External Malicious Code Identification
SI-02: Flaw Remediation
CP-09: System Backup}
CAV Internal
«Functional Part» SW Update Server
«Functional Part» CAV Central Gateway
«Functional Part» SW Update Client
«Data Object» SynchronizationRequest
«Data Object» RequestSoftwareUpdate
«Data Object» DeliverSoftwareUpdate
«Data Object» DeploySoftwareUpdate
«Data Object» AcknowlegdeSoftwareUpdate
«Data Object Flow»
Data_In
Data_Out

Fig. 2: Threat model / Pattern application

6 Conclusions and Future Work

The presented approach outlines a first step towards the overall automation of security-by-design according to ISO-21434. This intention can be realized, by the construction of so-called security patterns, stated in Section 2 and following the security pattern engineering process, presented in Section 4. Therefore, primary the architecture of an CAV needs to be modeled with the ARAM Framework and analyzed with the threat modeling tool ThreatGet to find potential threats. Consequently, the created pattern, can be applied to this very model, in the form of an extension for the ARAM Toolbox and help to mitigate occuring threats and risks in a certain scenario e.g., OTA update. Further research will focus on automating the remediation of cybersecurity risks using security patterns, the ARAM Framework, and the associated result validation and verification. In addition, it will be examined how the results of the security pattern engineering process on each layer of ARAM affect the required components on the layers below and how they can be automatically constructed and validated. One possibility is the utilization of software tools like LemonTree, developed by LieberLieber Software, which already enables automated validation of Systems Modeling Language (SysML) models. The future work aims in particular to incorporate vehicle functions and vehicle security development accurately, seamlessly, and with as little overhead as possible.

Acknowlegdements - The support for valuable contributions of Robert Bosch GmbH is gratefully acknowledged. The financial support by the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development and the Christian Doppler Research Association as well as the Federal State of Salzburg is also gratefully acknowledged.

References

1. Austmann, L.M.: Drivers of the electric vehicle market: A systematic literature review of empirical studies. Finance Research Letters **41**, 101,846 (2021)
2. Böhm, W., Broy, M., Klein, C., Pohl, K., Rumpe, B., Schröck, S. (eds.): Model-Based Engineering of Collaborative Embedded Systems : Extensions of the SPES Methodology. Springer, Cham, Switzerland (2021)
3. Chattopadhyay, A., Lam, K.Y., Tavva, Y.: Autonomous vehicle: Security by design. IEEE Transactions on Intelligent Transportation Systems **PP**, 1–15 (2020)
4. Conboy, K., Gleasure, R., Cullina, E.: Agile Design Science Research. In: B. Donnellan, M. Helfert, J. Kenneally, D. VanderMeer, M. Rothenberger, R. Winter (eds.) New Horizons in Design Science: Broadening the Research Agenda, pp. 168–180. Springer International Publishing, Cham (2015)
5. Draxler, D., Neureiter, C., Lastro, G., Schwartzkopff, T., Boumans, M.: A domain specific systems engineering framework for modelling electric vehicle architectures. In: 2019 IEEE Transportation Electrification Conference and EXPO Asia-Pacific. Jeju, Korea (2019)
6. Fitzpatrick, K.: How to write a security pattern. https://securitypatterns.io/ (2021), accessed: 2022-05-25

7. Force, J.T.: Security and privacy controls for information systems and organizations. Tech. rep., National Institute of Standards and Technology (2020)
8. Guillerm, R., Demmou, H., Sadou, N.: Engineering dependability requirements for complex systems - A new information model definition. In: 2010 IEEE International Systems Conference, pp. 149–152 (2010)
9. Haberfellner, R., Nagel, P., Becker, M., Büchel, A., von Massow, H.: Systems engineering. Springer (2019)
10. International Organization for Standardization: Iso 21434:2021 road vehicles - cybersecurity engineering (2021)
11. Karkare, P.: Model-based systems engineering for autonomous vehicle development. Tech. rep., Siemens PLM Software (2018)
12. Kim, K., Kim, J.S., Jeong, S., Park, J.H., Kim, H.K.: Cybersecurity for autonomous vehicles: Review of attacks and defense. Computers and Security **103**, 102,150 (2021)
13. Macher, G., Schmittner, C., Veledar, O., Brenner, E.: ISO/SAE DIS 21434 Automotive Cybersecurity Standard - In a Nutshell, pp. 123–135 (2020)
14. Mahmood, S., Nguyen, H.N., Shaikh, S.A.: Systematic threat assessment and security testing of automotive over-the-air (ota) updates. Vehicular Communications **35**, 100,468 (2022)
15. Martin, H., Ma, Z., Schmittner, C., Winkler, B., Krammer, M., Schneider, D., Amorim, T., Macher, G., Kreiner, C.: Combined automotive safety and security pattern engineering approach. Reliability Engineering and System Safety **198**, 106,773 (2020)
16. Möller, D.P., Haas, R.E.: Guide to Automotive Connectivity and Cybersecurity: Trends, Technologies, Innovations and Applications, 1st edn. Springer Publishing Company, Incorporated (2019)
17. Mwasilu, F., Justo, J.J., Kim, E.K., Do, T.D., Jung, J.W.: Electric vehicles and smart grid interaction: A review on vehicle to grid and renewable energy sources integration. Renewable and Sustainable Energy Reviews **34**, 501–516 (2014)
18. Neureiter, C.: A Domain-Specific, Model Driven Engineering Approach for Systems Engineering in the Smart Grid. MBSE4U - Tim Weilkiens, Hamburg, Germany (2017)
19. Pohl, K., Hönninger, H., Achatz, R., Broy, M. (eds.): Model-Based Engineering of Embedded Systems: The SPES 2020 Methodology. Springer, Heidelberg (2012)
20. Schmittner, C., Chlup, S., Fellner, A., Macher, G., Brenner, E.: Threatget: Threat modeling based approach for automated and connected vehicle systems. In: AmE 2020 - Automotive meets Electronics; 11th GMM-Symposium, pp. 1–3 (2020)
21. ThalesUK: Project resicav, d2.1 economic and technological feasibility of the cyres methodology. Tech. rep.
22. Varga, N., Bokor, L., Takács, A., Kovács, J., Virág, L.: An architecture proposal for v2x communication-centric traffic light controller systems. In: 2017 15th International Conference on ITS Telecommunications (ITST), pp. 1–7 (2017)
23. Ward, D., Wooderson, P.: Automotive Cybersecurity: An Introduction to ISO/SAE 21434, pp. i–xii (2021)
24. Weilkiens, T.: Systems Engineering mit SysML/UML, 3 edn. dpunkt, Heidelberg (2014)
25. Weilkiens, T., Lamm, J.G., Roth, S., Walker, M.: Model-Based System Architecture. John Wiley & Sons (2016)