

A cybersecurity operations approach for vehicles based on holistic monitoring

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

von der KIT-Fakultät für
Elektrotechnik und Informationstechnik
des Karlsruher Instituts für Technologie (KIT)

angenommene

DISSERTATION

von

M.Sc. Daniel Grimm

geb. in Karlsruhe

Tag der mündlichen Prüfung:

Hauptreferent:

Korreferent:

12.06.2026

Prof. Dr.-Ing. Eric Sax

Prof. Dr. Zsolt Szalay

Abstract

Vehicles evolve towards fully programmable, highly automated, and extensively connected systems. Yet, connectivity and the rise of software also pose the risk of cyberattacks. Each external interface can serve as an attack vector, and every line of code can introduce a vulnerability that could enable attackers to control vehicle functions remotely. Legal frameworks such as UN Regulation 155 and ISO/SAE 21434 have established cybersecurity engineering in the development process, but attackers continue to discover and exploit new vulnerabilities in vehicles. Considering cybersecurity risks only during development is therefore not sufficient. This dissertation shifts the focus to cybersecurity during operation. To keep vehicles secure throughout their lifecycle, monitoring, analysis, and response capabilities are required, and risk assessments must be continuously updated. The six key contributions (1)-(6) address the associated challenges of a procedural and technological nature.

A comparison of ISO/SAE 21434 with other cybersecurity processes reveals several gaps in cybersecurity during operation, particularly in incident handling. A new operational process (1) is proposed to improve ISO/SAE 21434, especially for the workflow of incident and vulnerability handling. The mandatory risk analysis conducted during development has, to date, not taken cybersecurity monitoring into account. Instead of implementing monitoring measures in the vehicle without methodological support, this thesis proposes risk-focused monitoring (2). Suitable requirements are defined in the risk analysis to enable early detection of critical attack paths. Furthermore, by integrating formalized knowledge about attacker behavior and resulting observables, the approach allows to methodically select appropriate measures.

Technologically, related work focuses on network-centric measures, particularly intrusion detection systems (IDS). Assessing the vehicles' cybersecurity situation is therefore limited to isolated information, such as IDS alerts. In contrast, this work proposes a holistic approach that includes the vehicle's operational context (3), collecting information from the vehicles over the air (4) in a Vehicle Security Operations Center (VSOC), where a Knowledge Graph (KG) with a comprehensive data model links in-vehicle information with development documents and the operating process (5). Connecting technical information to vehicle variants, operational context, the vehicle network, known risks, vulnerabilities, and past incidents, establishes the necessary background to quickly and thoroughly assess whether an incident exists and what threats it poses.

Monitoring and analysis ultimately strive to identify appropriate responses. While updates are the obligatory final solution, their development may take months. As safety-critical threats may require rapid containment, pre-homologation and over-the-air activation of incident responses (6), such as disconnecting the vehicle's internet connection, provide an immediate and targeted response capability.

Future work can build on these contributions in two key directions. The volume and structure of data in the KG offer diverse opportunities for artificial intelligence (AI) to reduce analyst workload, with additional challenges specific to safety-critical and variant-rich systems such as trust in AI decisions. Furthermore, capturing the operational context for cybersecurity lays the groundwork to integrate safety and cybersecurity monitoring. A shared contextualization of insufficiencies, security issues, and failures, is currently underexplored but particularly important for highly automated vehicles.

Since software and connectivity have become indispensable in vehicles, cybersecurity remains a key area of research, as attackers will always seek to identify the weakest point to exploit. Overall, the outcomes of this work provide a foundation for securing vehicles not only at the time of their release, but throughout their entire lifecycle in the face of the ever-evolving threat landscape.

Kurzfassung

Fahrzeuge entwickeln sich zu vollständig programmierbaren, hochautomatisierten und umfassend vernetzten Systemen. Doch Vernetzung und Softwarewachstum bergen auch das Risiko von Cyberangriffen. Jede Schnittstelle kann als Angriffsvektor dienen, und jede Zeile Code kann eine Schwachstelle darstellen, die Angreifern die Fernsteuerung von Fahrzeugfunktionen ermöglicht. Rechtliche Rahmenwerke wie die UN-Regelung 155 und ISO/SAE 21434 haben Cybersicherheit im Entwicklungsprozess etabliert, doch Angreifer entdecken und nutzen weiterhin neue Schwachstellen aus. Es reicht daher nicht aus, Cybersicherheitsrisiken nur während der Entwicklung zu berücksichtigen. Diese Dissertation legt den Fokus auf die Cybersicherheit während des Betriebs. Um die Sicherheit von Fahrzeugen über ihren gesamten Lebenszyklus hinweg zu gewährleisten, sind Überwachungs-, Analyse- und Reaktionsfähigkeiten erforderlich, und Risikobewertungen müssen kontinuierlich aktualisiert werden. Die sechs wesentlichen Beiträge (1)–(6) befassen sich mit den damit verbundenen prozessualen und technologischen Herausforderungen.

Ein Vergleich von ISO/SAE 21434 mit anderen Cybersicherheitsprozessen zeigt mehrere Lücken in der Cybersicherheit während des Betriebs auf, insbesondere beim Umgang mit Vorfällen. Ein neuer Betriebsprozess (1) wird vorgeschlagen, um diese zu schließen. Die obligatorische Risikoanalyse während der Entwicklung hat bislang die Cybersicherheitsüberwachung nicht berücksichtigt. Diese Arbeit schlägt daher eine risikoorientierte Überwachung (2) vor. Geeignete Anforderungen werden bereits in der Risikoanalyse definiert, um eine frühzeitige Erkennung kritischer Angriffspfade zu ermöglichen. Die Integration formalisierten Wissens über das Verhalten von Angreifern und daraus resultierende Beobachtungsgrößen erlaubt die methodische Auswahl geeigneter Maßnahmen.

Technologisch konzentrieren sich verwandte Arbeiten auf netzwerkzentrierte Maßnahmen wie Intrusion Detection Systeme (IDS). Die Bewertung der Cybersicherheitslage der Fahrzeuge beschränkt sich daher auf isolierte, technische Informationen. Diese Arbeit schlägt einen ganzheitlichen Ansatz vor, der den Betriebskontext des Fahrzeugs (3) einbezieht, Informationen drahtlos (4) in einem Vehicle Security Operations Center (VSOC) sammelt und dort einen Knowledge Graph (KG) mit umfassendem Datenmodell einsetzt (5). Die Verknüpfung technischer Informationen mit Fahrzeugvarianten, dem Betriebskontext, dem Fahrzeugnetzwerk, bekannten Risiken, Schwachstellen und vergangenen Vorfällen schafft die Grundlage, um schnell zu beurteilen, ob ein Vorfall vorliegt und welche Bedrohungen davon ausgehen.

Überwachung und Analyse zielen letztendlich darauf ab, geeignete Gegenmaßnahmen zu ermitteln. Zwar sind Software-Updates die unverzichtbare endgültige Lösung, doch kann ihre Entwicklung Monate dauern. Da sicherheitskritische Bedrohungen eine rasche Eingrenzung erfordern können, ermöglichen die Vorab-Typgenehmigung und drahtlose Aktivierung von Reaktionsmaßnahmen (6), wie die Unterbrechung der Internetverbindung, eine sofortige und gezielte Reaktion.

Zukünftige Arbeiten können auf diesen Beiträgen in zwei wesentlichen Richtungen aufbauen. Die Datenmenge und -strukturierung im KG bieten vielfältige Möglichkeiten für künstliche Intelligenz (KI), den Arbeitsaufwand der Analysten zu verringern, wobei spezifische Herausforderungen wie das Vertrauen in KI-Entscheidungen bestehen. Darüber hinaus schafft die Erfassung des Betriebskontexts die Grundlage für die Integration von Safety- und Cybersicherheitsüberwachung. Eine gemeinsame Kontextualisierung von Unzulänglichkeiten, Cybersicherheitsvorfällen und Ausfällen ist derzeit noch wenig erforscht, aber für hochautomatisierte Fahrzeuge besonders wichtig.

Nachdem Software und Vernetzung in Fahrzeugen unverzichtbar geworden sind, bleibt die Cybersicherheit ein zentraler Forschungsbereich, da Angreifer stets versuchen werden, Schwachstellen zu finden und auszunutzen. Die Ergebnisse dieser Arbeit bilden eine Grundlage für die Absicherung von Fahrzeugen nicht nur bei ihrer Einführung, sondern über ihren gesamten Lebenszyklus.

Declaration on the use of generative AI

During the creation of this thesis, tools based on generative AI were used. In accordance with the guidelines on the use of generative AI issued by the KIT Faculty of Electrical Engineering and Information Technology¹, a declaration on the use of these tools is provided: In the preparation of this thesis, I used Grammarly for grammatical and stylistic revision. Claude was used for stylistic revision and formulation support. ChatGPT was used for stylistic revision, formulation support, and generating code and database queries in the prototypical implementation. DeepL was used for translating text passages and for stylistic revision. After using these tools, I reviewed and revised the content, and I assume full responsibility for the content of this publication.

¹ https://www.etit.kit.edu/rd_download/Handout_GenerativAI_ETITKIT.pdf

Preface

This dissertation was conducted during my time as a researcher at the Karlsruhe Institute of Technology (KIT) at the Institut für Technik der Informationsverarbeitung (ITIV), where I had the opportunity to work with many wonderful people. This experience significantly shaped and advanced me both personally and professionally.

First, I would like to thank Prof. Dr.-Ing. Eric Sax for supervising my thesis and for his continuous support and trust throughout the many years I was given the opportunity to spend at ITIV, which gave me the freedom to shape my own research direction. I would also like to thank Prof. Dr. Zsolt Szalay for serving as the second examiner of this thesis and for supporting a research stay at the Budapest University of Technology and Economics, which contributed significantly to the success of this work. Many thanks also to the other members of the examination committee, consisting of Prof. Dr.-Ing. Mike Barth, Prof. Dr. Jasmin Aghassi-Hagmann, and Prof. Dr.-Ing. Mathias Noe, who took the time for preliminary discussions and ensured a comfortable examination atmosphere.

My special thanks go to my colleagues at ITIV, without whom this dissertation certainly would not have been possible. In addition to providing inspiration and opportunities for constructive discussions, they ensured above all that the work was enjoyable and that my time at ITIV was also very enriching on a personal level. First and foremost, I would like to thank Marc Schindewolf, David Kraus, Moritz Zink, and Marco Stang, who played a crucial role for me, both academically and beyond. I would also like to thank my fellow doctoral researchers from FZI and from industry, who broadened my perspective and were another source of fun and inspiration along the way.

I would also like to thank my family for their backing and support during this time. It's great to know that I can always count on you. My final and greatest thanks go to Carolin Autenrieth, for always being there for me, for your patience, understanding, unlimited support, and faith in me throughout this journey, without which this work would not have been possible.

Karlsruhe, July 2026

Daniel Grimm

Table of contents

Abstract	i
Kurzfassung	iii
Preface	vii
1 Initial situation and objectives	1
1.1 Motivation	1
1.2 UN Regulation 155	6
1.3 Situation	8
1.4 Implications	11
1.5 Thesis objectives	13
1.6 Structure of this work	13
2 Terminology, technology and development process	15
2.1 Building blocks of future E/E architectures	15
2.1.1 Centralized, powerful network architecture	15
2.1.2 Abstraction with software architecture	18
2.1.3 Virtualization	22
2.1.4 Orchestration	24
2.2 Automotive development lifecycle	25
2.2.1 Lifecycle models	25
2.2.2 Model-based systems engineering	27
2.2.3 Variant and release management	30
2.2.4 Update development and deployment	32
2.2.5 Safety	34
2.3 ISO/SAE 21434 Cybersecurity Engineering	39
2.3.1 Development processes	40

2.3.2	Processes for cybersecurity operations	48
3	Ensuring vehicular cybersecurity during operations	53
3.1	Idea of this thesis	53
3.1.1	Processes	54
3.1.2	Technologies	55
3.2	Requirements	58
4	Current state of cybersecurity operations for vehicles	61
4.1	Processes	63
4.1.1	Security Operations Centers	63
4.1.2	Automotive cybersecurity operations approaches	66
4.1.3	IT security standards and guidelines	70
4.1.4	Procedures to specify monitoring requirements	73
4.2	Technologies	75
4.2.1	Cyber situation awareness	75
4.2.2	Monitoring	76
4.2.3	Cybersecurity information management	87
4.2.4	Incident response	92
5	Discussion of the current state of automotive cybersecurity operations	97
5.1	Processes	97
5.1.1	Process(es) for cybersecurity operations activities	97
5.1.2	Procedure to specify monitoring requirements	103
5.2	Technologies	104
5.2.1	In-vehicle monitoring of cyber and physical aspects	104
5.2.2	Fleet-wide collection	106
5.2.3	Connection with additional information	106
5.2.4	Immediate response over the air	114
5.3	Summary	114
5.3.1	Processes	114
5.3.2	Technologies	115

6	Conception of a cybersecurity operations approach for vehicles	119
6.1	Processes	121
6.1.1	Process(es) for cybersecurity operations activities	121
6.1.2	Procedure to specify monitoring requirements	122
6.2	Technologies	123
6.2.1	In-vehicle monitoring of cyber and physical aspects	124
6.2.2	Fleet-wide collection	125
6.2.3	Connection with additional information	126
6.2.4	Immediate response over the air	127
7	Improving ISO/SAE 21434 for cybersecurity operations	131
7.1	Terminology: events, incidents and indicators	132
7.2	Operations process for a VSOC	134
7.2.1	Preparation	134
7.2.2	Cybersecurity monitoring	135
7.2.3	Triage	136
7.2.4	Vulnerability Handling	137
7.2.5	Incident Handling	139
7.2.6	Closure and post-analysis of vulnerability and incident handling	140
7.2.7	Further VSOC capabilities	142
7.3	Further continuous activities beyond the VSOC	143
7.3.1	TARA management	143
7.3.2	Support organization	146
7.3.3	Update development	147
7.3.4	Continuous improvement	147
8	A threat information-based procedure to specify cybersecurity monitoring requirements	149
8.1	TARA with threat information	151
8.1.1	Threat scenario and damage scenario identification	152
8.1.2	Attack path modeling	154
8.1.3	Risk treatment decision and derivation of monitoring goals	156
8.2	Monitoring concept definition	158
8.2.1	Identification of relevant observables and observations	158

8.2.2	Mapping the capabilities of available sensors	161
8.2.3	Establishing the maximal monitoring requirements	162
8.2.4	Optimization and allocation of requirements	163
9	Holistic cybersecurity monitoring of vehicle fleets	165
9.1	Overview of the information considered in holistic monitoring . . .	167
9.1.1	Vehicle fleet	168
9.1.2	Development process	170
9.1.3	Support organization	174
9.1.4	Operations process	174
9.2	In-vehicle monitoring of cyber and physical aspects	176
9.2.1	Network security monitoring	176
9.2.2	Operational context monitoring	180
9.3	Fleet-wide collection	184
9.3.1	Centralized buffering and filtering	184
9.3.2	Secure channel for monitoring data	187
9.4	Connection with additional information in the VSOC	188
9.4.1	Labeled Property Graphs as a Knowledge Graph of cybersecurity information	189
9.4.2	Information preprocessing and integration in the Knowledge Graph	190
9.4.3	The ASCOT data model	192
9.4.4	Potentials for the Use of Artificial Intelligence	203
10	Immediate response over the air	207
10.1	Pre-homologated responses	208
10.1.1	Analysis and categorization of response actions	208
10.1.2	Response actuators	210
10.2	Response activation over-the-air	211
10.2.1	Analysis and decision	212
10.2.2	Course of action preparation	213
10.2.3	Secure response transmission	213
10.2.4	Setting the course of action	214
10.2.5	Execution	215

11 Prototypical implementation of monitoring and response approaches	217
11.1 Example item: Adaptive headlight system	218
11.2 Technical platforms and vehicle variants	219
11.2.1 Miniature vehicle	219
11.2.2 Simulated E/E architecture and fleet synchronization	221
11.2.3 Realized vehicle variants and capabilities	223
11.3 Applying the procedure to specify cybersecurity monitoring requirements	226
11.3.1 Threat analysis and risk assessment with threat information	226
11.3.2 Monitoring concept definition	231
11.4 Holistic monitoring of the adaptive headlight system	235
11.4.1 Monitoring technology	235
11.4.2 Further in-vehicle software	240
11.4.3 Fleet data collection in the Knowledge Graph	241
11.5 Demonstration of the incident handling workflow	249
11.5.1 Implemented attack path	249
11.5.2 Using the Knowledge Graph to analyze the attack	254
11.6 Discussion of the outcomes and limitations	260
12 Summary and future work	265
12.1 Thesis contributions	266
12.2 Outlook	269
A Further details on the outcomes of this work	273
A.1 Terminology of the resulting process	273
A.2 The ASCOT threat catalog	281
A.3 The ASCOT data model	292
A.4 Response actions	296
B Exemplary application of the procedure to specify monitoring requirements	297
C Further background	305
C.1 Networking technologies	305
C.2 Software architecture	306

C.3 Security measures in vehicles 310

Acronyms 315

Glossary 323

List of Figures 333

List of Tables 337

Bibliography 339

 Own publications 339

 References 343

 Standards and public guidelines 380

1 Initial situation and objectives

1.1 Motivation

Vehicles are undergoing a fundamental transformation. For more than a century since the invention of the automobile¹ the mechanical parts and combustion engines were the main drivers for innovation. However, today the car's software has taken this leading role [SNI23]. The central trends, automated driving, connectivity, electrification of the powertrain, regionalization of customer requirements, and mobility on demand, are disrupting the whole industry from the OEM² to the suppliers [Mog+24]. More and more, what was differentiated through mechanical and electronic properties is becoming the SDV³. While software was statically combined with microcontrollers as ECU⁴ in the past, the software architecture is becoming increasingly dynamic nowadays. At the core are generalized, high-performance computing platforms with multiple microprocessors, IT⁵-derived OS⁶, virtualization and middlewares, that enable flexible execution and connection of applications.

Currently, up to 150 hosts in vehicles [Sla24], including ECUs and generalized computing platforms, are connected via networks of different technologies. These include bus systems such as CAN⁷, though Ethernet is gaining ground [Pol24]

¹ Carl Benz' *Patent-Motorwagen* went on its first test run for the public on July 3, 1886.

² Original Equipment Manufacturer, synonym for vehicle manufacturers

³ software-defined vehicle

⁴ Electronic Control Units

⁵ Information Technology (cf. glossary, page 326)

⁶ Operating System

⁷ Controller Area Network

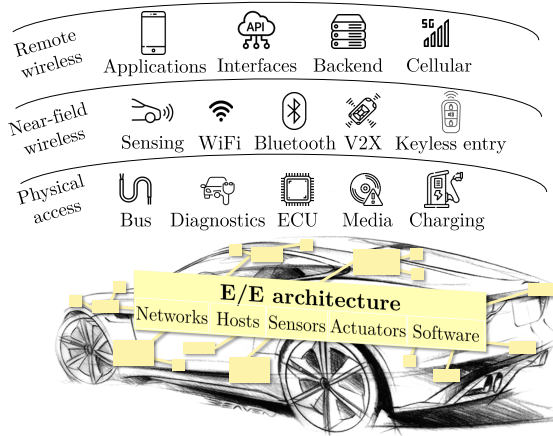


Figure 1.1: The connected car and its interfaces serving as attack vectors¹¹.

(cf. Section 2.1.1 on page 15). Different sensors observe the vehicle’s environment for driver assistance systems, the first step to automated driving. The hosts, sensors and actuators, and the networks make up the vehicle’s network architecture. Together, the network architecture and the software are the E/E architecture⁸ - the heart of modern vehicles (cf. yellow in Figure 1.1 on page 2).

With the increase in software-based functions, the E/E architecture had to grow, which drew attention to SOA⁹ (cf. Section 2.1.2 on page 18), as it offers better maintainability and reusability compared to a static, signal-based software architecture [Kuk+22]. Moreover, MBSE¹⁰ was established to handle the growth and the distributed development process with suppliers, while ensuring strict safety requirements (cf. Section 2.2.2 on page 27). Another aspect of change in E/E architectures, alongside strong growth in software, is the trend toward more centralization, such as in zone-based network topologies (cf. Section 2.1.1).

⁸ electrical and electronical architecture

⁹ Service-oriented Architecture (cf. glossary, page 329)

¹⁰ model-based systems engineering

Wireless and wired interfaces connect the vehicle to the outside world (cf. upper part of Figure 1.1 on page 2). Cellular networks (3G, 4G, 5G) and WiFi serve to transmit information to internet-connected servers, or receive OTA¹² software updates. OTA updates are important to provide new features or to improve functions [SNI23]. For example, over 50 electronic components of the Mercedes S-Class (W223) can be updated remotely [Sch+20a]. Novel competitors, including Rivian, Tesla and Polestar, provide OTA updates for most of the components [Fis23]. Servers for the vehicles operation, such as for OTA updates, are called *backend* systems.

Although sensors, computing capabilities, and connectivity bring great customer benefits, the vehicle is no longer a completely enclosed system. Thus, the automotive industry faces the downside of expansion lately: the risk of cyberattacks and data theft. Vehicular cybersecurity - developing, producing, and operating vehicles that are free from unnecessary cyber threats - is now imperative.

Definition 1.1: Road vehicle cybersecurity

ISO¹³/SAE¹⁴21434 defines cybersecurity as the "condition in which assets are sufficiently protected against threat scenarios to items of road vehicles, their functions and their electrical or electronic components." [ISO21] (cf. Section 2.3 on page 39 and Definition 2.20 on page 43)

Cybersecurity is challenged by all central trends, not only connectivity. Cybersecurity concerns impact trust in automated driving, as the majority of consumers surveyed wanted to understand how self-driving cars are secured [Boo+23]. Electrification adds an external interface (charging) and distinct threats (financial fraud, high voltages). Also, a customer tampering with a shared vehicle can

¹¹ Car: "The C-X16 3.0L V6 Supercharged Hybrid Sports Car." by jaguarmena, Wikimedia Commons. Icons (www.flaticon.com): 5G, backend by pixel-perfect, API by Cap Cool, V2X by pongsakornRed, Wi-Fi, charging, smartphone by freepik, diagnostic, keyless by bsd, Bluetooth by prettyicons, bus by Zaenul Yahya, Media by Smashicons, ECU by monkik.

¹² over-the-air

¹³ International Organization for Standardization

¹⁴ Society of Automotive Engineers

be an attack vector on future customers or the backend systems, especially for automated vehicles.

While software is indispensable, every line of code increases the risk for introducing weaknesses that could become entry points for attacks. From the early 2000s, researchers started to investigate vehicular cybersecurity (e.g. [HKD08b; Kos+10; WWP04; BB07]). Until today, a manifold of attack vectors (cf. upper part of Figure 1.1 on page 2 and Figure 1.2 on page 4) has been exploited by white-hat as well as black-hat (i.e. actors with malicious intent) hackers. White-hat hackers try to find vulnerabilities and weaknesses with good intent (informing the manufacturer) to identify previously unknown threats. They usually demonstrate the feasibility of an attack, to verify the compromised security properties.

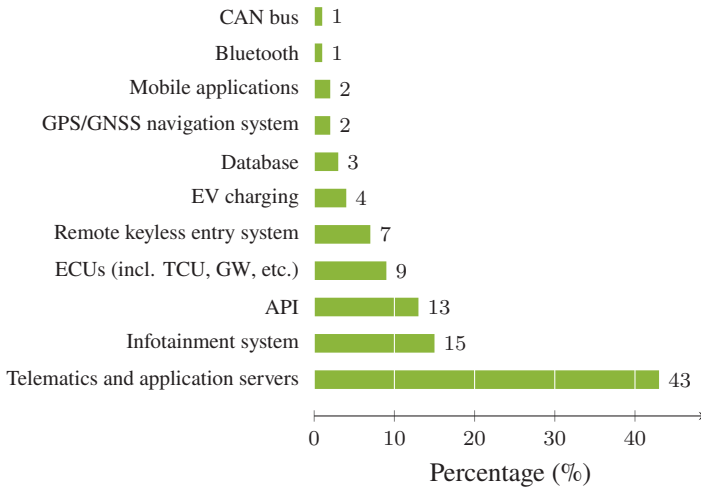


Figure 1.2: Distribution of attack vectors for incidents in automotive systems in 2023, based on [Ups24b].

Researchers found that the internal communication and software of vehicles can be manipulated with physical access to the vehicle via the OBD¹⁵ interface or by attacks via short-range communication (e.g. Bluetooth) [Kos+10] (cf. innermost and middle layer of external interfaces in Figure 1.1 on page 2). Yet, the CAN bus was the research's focus in the beginning, as it lacks inherent cybersecurity mechanisms. Ultimately, the "Jeep Hack" [MV14] drew game-changing attention to the problem [Gre15; Gual15], which led to a recall of 1.4 million cars and estimated costs of US\$ 600 million [Dei+19]. Beyond financial and reputational damage, such attacks carry direct safety implications as a vehicle whose functions can be remotely manipulated poses a physical risk to its occupants and other road users. Initially, it was mainly white hats that attracted interest in cybersecurity, but in 2018, for the first time, most of the incidents were associated with black hats [Ups19]. At the same time, the focus shifted from the CAN bus towards scalable attack vectors (cf. Figure 1.2).

Remote attacks, e.g. via the backend or the user's smartphone, are particularly critical because they could put an entire fleet of vehicles out of operation at the same time [Che+11; Sil+17]. In this thesis, the term fleet usually refers to all vehicles of a manufacturer. In fact, in 2023 around 80% of all real-world incidents analyzed by the company Upstream Security were long-ranging remote attacks [Ups24b]. Among these, the most frequently used attack path involved telematics and application servers (cf. Figure 1.2 on page 4). As these backend systems are part of the IT server infrastructure of the OEM, this highlights the connection of IT system cybersecurity to the cybersecurity of vehicles.

In general, the number of published incidents increased by a factor of five over the last years, from 60 in 2018 [Ups19] to 295 in 2023 [Ups24b]. In 2023, according to an industry survey, 14% of the automotive organizations experience cyber incidents impacting the products monthly [Min23a]. In this work, the term is clarified (cf. Section 7.1 on page 132), resulting to the following definition:

¹⁵ On-Board Diagnostics (cf. glossary, page 327)

Definition 1.2: Cybersecurity incident

An occurrence in a system that involves a violation of product cybersecurity, product safety or organizational cybersecurity policy.

Definition 1.3: Cyber attack

A (cyber) attack is defined as an "attempt to destroy, expose, alter, disable, steal or gain unauthorized access to or make unauthorized use of an asset" [ISO18a].

An incident with malicious intent is, therefore, the effect of an attack. For example, asking a colleague on long-term sick leave for her login credentials to work documents for business purposes is not an attack, as the intent is not malicious. However, it may be an incident if the company policy prohibits it.

1.2 UN Regulation 155

Due to these risks and increase in incidents, legislation has moved to make the cybersecurity of vehicles mandatory. The UN R155¹⁶ [EEC21a], that came into force in January 2021, was developed by the World Forum for Harmonization of Vehicle Regulations (WP.29), which is a working party of the Sustainable Transport Division of the United Nations Economic Commission for Europe (UNECE). It applies to its 64 member countries, e.g. the European Union, but other countries define their own similar regulations¹⁷. UN R155 contains twelve chapters and five annexes, that majorly cover the implementation of a CSMS¹⁸ and compliance with certain requirements for the approval of new vehicle types, whereby a certified CSMS is a prerequisite for type approval.

¹⁶ UN Regulation No. 155

¹⁷ USA, Canada, China

¹⁸ Cyber Security Management System

Definition 1.4: Cyber Security Management System

UN R155 [EEC21a] defines CSMS as "a systematic risk-based approach defining organisational processes, responsibilities and governance to treat risk associated with cyber threats to vehicles and protect them from cyber-attacks"

Chapter 7 of UN R155 describes in detail the requirements for the CSMS and for type approval [EEC21a, Chapter 7.2 and 7.3]. First of all, the CSMS must cover all lifecycle phases: development, production, and post-production phase¹⁹. Accordingly, the CSMS must include processes providing risk management, effective cybersecurity measures, testing, monitoring, detection, response to attacks, and supplier management. For the approval of new types, the CSMS must be certified, and its defined processes must be properly implemented, with particular reference to the threats and proposed mitigations from Annex 5 of UN R155.

Definition 1.5: Cybersecurity measure

ISO/IEC 27000 [ISO18a] and the automotive standard ISO/SAE 21434 [ISO21] define a cybersecurity control to be a "measure that is modifying risk". ISO/IEC 27000 stipulates that such measures can be technical (software, hardware) and non-technical (process, policy, practice, action). In the sense of this thesis, the term cybersecurity measure refers to technical measures, unless indicated differently.

Measures of the vehicle manufacturer must include aftermarket software and applications (cf. §7.3.5 [EEC21a]). Thus, more or less, the manufacturer is liable for potentially unknown external components. Beyond the development, vehicle manufacturers and suppliers are now obliged to consider how the vehicle's cybersecurity can be *ensured in operation*. To this end, this thesis defines the term *cybersecurity operations*:

¹⁹ cf. glossary, page 328

Definition 1.6: Cybersecurity operations

The processes and procedures for maintaining the cybersecurity of a system after its production until the end of cybersecurity support.

In detail, UN R155 mandates to implement measures that provide and support monitoring, detection, response, and forensic capabilities. Also, annual reports on monitoring activities are required, in particular on new cyberattacks, as well as reports on the effectiveness of the implemented mitigations.

Definition 1.7: Cybersecurity monitoring (cf. Definition 7.4)

The process of collecting, pre-processing and formatting cybersecurity information for subsequent processing.

Definition 1.8: Response

A cybersecurity response is a cybersecurity control (cf. Definition 1.5 on page 7, technical and/or non-technical) that modifies the risk of a system during its operation. A synonym for cybersecurity response used throughout this work is reaction.

Responses are implemented when the estimation of prevailing risks changes, which can be induced by incidents, new threats or vulnerabilities. It is not mandatory to respond to *all* changed risks. As in development (cf. Section 2.3.1 on page 40) risk treatment can also accept, share or transfer the risk.

1.3 Situation

Since July 2022, new vehicle types no longer receive approval (homologation) in member countries of UN R155 if the respective requirements were not considered. Accordingly, first new vehicle types complying with this regulation have been homologated [Eic23]. From July 2024, the regulation applies to all newly produced

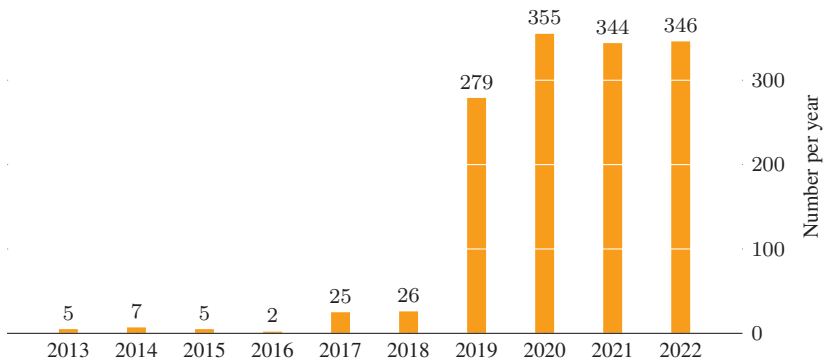


Figure 1.3: Number of new automotive security vulnerabilities 2013-2022 (CVE entries), based on [Vic23].

vehicles - not limited to new vehicle types. Indeed, this led to discontinuation of car models by several manufacturers [C2A24].

However, the industry already sees itself on the way to mature cybersecurity: 40% of the 140 participants of a survey in 2023 report their organization's maturity as "established" [Min23a] and cybersecurity is now treated as required aspect of quality [Eic23], which is a positive trend. 71 % of the participants in the survey say, their organization has a cybersecurity policy in place [Min23b]. At the same time, however, the number of officially published vulnerabilities in the CVE²⁰ between 2018 and 2022 increased by a factor of 10 (cf. Figure 1.3 on page 9). This reflects either a change in culture (responsible disclosure policy, and promoting sharing of knowledge), or at least an increased awareness of cybersecurity. Considering cybersecurity as a requirement at all was a significant step forward from the prevailing "security by obscurity," i.e., if the vehicle's architecture is secret, nobody will know its vulnerabilities or how to fiddle around with it. In the short term, obscurity may deter hackers, because it takes effort to understand a vehicles' network. In the long term, cybersecurity can only be

²⁰ Common Vulnerabilities and Exposures (cf. glossary, page 324)

guaranteed if the risks of attacks on vehicles are reduced in a structured manner, as stated in UN R155.

Therefore, risk management and assessment is a core element throughout the life-cycle. Analogous to the risk-based prioritization of functional safety (e.g., ISO 26262 [ISO18b]), TARA²¹ methods are becoming important tools for cybersecurity. TARA is promoted in particular in the ISO/SAE 21434 [ISO21] standard to derive the specification of cybersecurity requirements (cf. Section 2.3.1 on page 40). UN R155 recognizes ISO/SAE 21434 as highly supportive for implementing its requirements [Eur21], which accounts for its industrial relevance²².

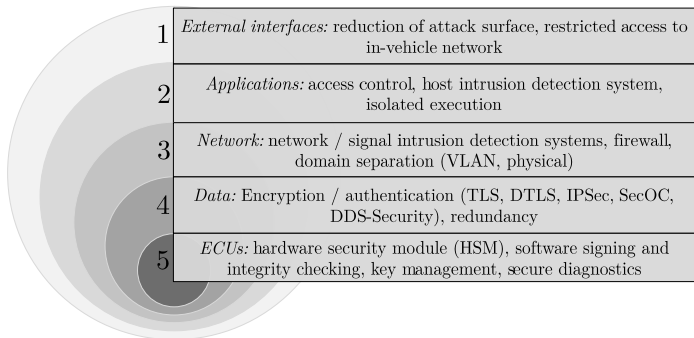


Figure 1.4: Cybersecurity measures in the vehicle in a defense in depth approach (derived from [Web19; Web18]).

On the technological side, IT cybersecurity measures were adapted, and automotive-specific ones were introduced [Rum+20]. A defense-in-depth approach is recommended, combining several layers from external interfaces to the ECU’s software [ISO21] (cf. Figure 1.4 on page 10 and Section C.3 on page 310). These

²¹ Threat Analysis and Risk Assessment

²² ISO/SAE 21434 was the most important standard for the industry in 2023 [Min23a]

include, for example, firewalls, VLAN²³, encryption using TLS²⁴ or IPSec²⁵, digital signatures and certificates, hash-based message authentication, access control, IDS²⁶ or IPS²⁷. Particular automotive-specific approaches are SecOC²⁸ to authenticate CAN communication, and the secured diagnostic access with UDS²⁹. Besides, specific hardware accelerates cryptographic operations, stores cryptographic material, limits physical network and host access, and constitutes the root of trust for secure software execution with signature and integrity checking. OTA updates are another important technical feature. If a vulnerability needs to be eliminated, this capability can limit the number of recalls to workshops for cybersecurity reasons.

1.4 Implications

Given these processes and technologies, one might argue that vehicles are sufficiently secured. But what if one considers the entire lifecycle, not just the threats known during development? The number of new vulnerabilities that become known every year³⁰ shows that it is practically impossible to avoid new threats. 45% of the participants of the aforementioned survey say, that their organization has dedicated roles for cybersecurity operations, and the scope of monitoring (i.e., amount of sources considered) is seen as a distinguishing factor for the maturity of cybersecurity engineering [Min23b]. However, compared to the 71% having cybersecurity policies, there exists a gap and potentially required investments in this field. Also, developing and releasing updates involves several challenges,

²³ Virtual Local Area Networks (cf. glossary, page 330)

²⁴ Transport Layer Security (cf. glossary, page 329)

²⁵ Internet Protocol Security

²⁶ Intrusion Detection System (cf. glossary, page 326)

²⁷ Intrusion Prevention System (cf. glossary, page 326)

²⁸ Secure Onboard Communication (cf. glossary, page 328)

²⁹ Unified Diagnostic Services (cf. glossary, page 329)

³⁰ e.g., only in 2022 there were 346 new vulnerabilities in automotive components and services [Vic23]

including safety and variant management [Hen+23], limiting the pace of fixing vulnerabilities.

At this point, it is worth reflecting on the differences between safety and security. Functional safety protects people and the environment from potentially dangerous vehicles. Cybersecurity, on the other hand, protects vehicles from people, e.g., from being stolen, tampered with, or from safety functions being disabled. While safety protects against failures of known things ("our" components), cybersecurity shall protect the vehicle from unknown and malicious behavior of people. If failure estimations are correct today, they will most likely be tomorrow. If a component is *thought to be* secure today, an attacker may discover a new vulnerability tomorrow.

Hence, a one-time risk assessment during development, resulting in preventive measures is not sufficient, because the (known) risks change over time. Cybersecurity must be ensured continuously from the development to decommissioning. These thoughts and the legally mandatory requirements of UN R155 yield the following key cybersecurity tasks an OEM has during the operational phase:

- T1 Perform monitoring to provide data for analysis, including vehicle data
- T2 Analyze and detect threats, vulnerabilities and attacks
- T3 Keep risk assessments up to date
- T4 Keep cybersecurity measures effective
- T5 Respond within a reasonable timeframe, if required

In essence, this dissertation addresses the central question: What is necessary to ensure long-term vehicle cybersecurity, especially during operation, and adequately counter (new) risks?

1.5 Thesis objectives

This work focuses on ensuring the cybersecurity of vehicles during operations. As the legislation has stipulated, monitoring and response capabilities are required. On the one hand, this requires the appropriate organizational processes to make well-considered decisions quickly. But it also requires the right technologies in the vehicle and beyond, to adapt to the changing risks over the lifecycle.

Research Question 1: Process

What phases and steps should a cybersecurity engineering process include to support the cybersecurity operations of a vehicle fleet?

Research Question 2: Monitoring

What data is helpful to assess the cybersecurity risks during operation? How can this data be collected, pre-processed and formatted to support cybersecurity operations?

Research Question 3: Response

How can responses be enabled that restore the cybersecurity of the vehicle fast enough when new risks are identified? What is the role of OTA updates in this context?

1.6 Structure of this work

The following chapter familiarizes the reader with E/E architectures and development processes (cf. Chapter 2 on page 15). Afterward, the vision of this work for keeping vehicles secure over their lifecycle is introduced (cf. Chapter 3 on page 53). Possible approaches from a technological and procedural perspective from science and industry are then presented (cf. Chapter 4 on page 61) and discussed (cf. Chapter 5 on page 97). Based on this, the solution concept is outlined

(cf. Chapter 6 on page 119), which is detailed and designed further on (cf. Chapter 7 on page 131, Chapter 8 on page 149, Chapter 9 on page 165 and Chapter 10 on page 207). The prototypical implementation (cf. Chapter 11 on page 217) focuses on a particular example system, including realizing an attack path, where processes and technological aspects (monitoring and response) are applied. The thesis concludes with a summary of the contributions and a discussion of the resulting new research perspectives (cf. Chapter 12 on page 265).

2 Terminology, technology and development process

2.1 Building blocks of future E/E architectures

2.1.1 Centralized, powerful network architecture

Technologies

The network architecture evolution, including high-bandwidth physical layers and more centralized topologies is the basis for the SDV. Since the introduction of the CAN bus in the 1980's to reduce wiring while maintaining reliability, various new physical layer technologies found application in the vehicle, including LIN¹, SerDes², USB³, FlexRay and Ethernet. With different costs and bandwidths, they all fulfill certain requirements. A qualitative comparison indicates several competing technologies around 10 MBit/s and above 1 GBit/s (cf. Figure 2.1 on page 16). This has led to hybrid network architectures, consisting of multiple bus systems plus point-to-point switched Ethernet. For example, Mercedes' STAR3 architecture features at least⁴ 13 (+1 diagnostic) CAN buses, five point-to-point HSVL⁵, a FlexRay, and an Ethernet (100BASE-T1) backbone. While special-purpose technologies may yield direct cost benefits, they introduce other

¹ Local Interconnect Network

² Serializer/De-Serializer (cf. glossary, page 329)

³ Universal Serial Bus

⁴ cf. schematic overview, Figure 2.3 on page 19.

⁵ High Speed Video Link (cf. glossary, page 325)

challenges compared to a homogeneous (e.g. Ethernet- or CAN-only) network, such as tooling [MH21].

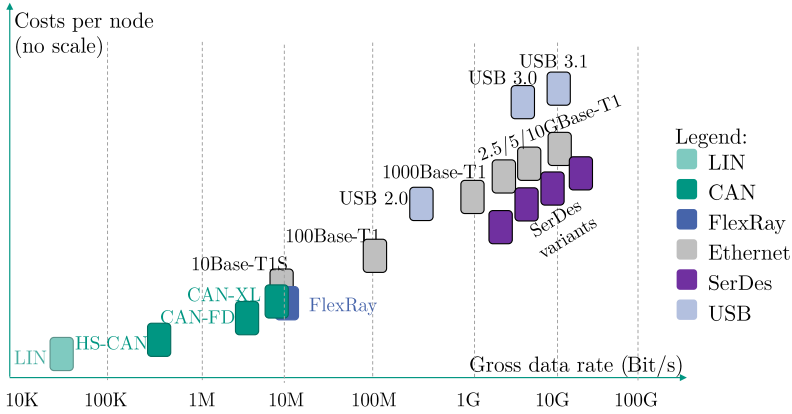


Figure 2.1: Qualitative cost comparison of automotive networking technologies (based on [MH21]).

High-bandwidth technologies like Ethernet are important for data-intensive applications such as automated driving and infotainment. But Ethernet, in particular, allows for more extendibility, flexible topologies (cf. Section *Topologies* on page 17), and SOA (cf. Section 2.1.2 on page 18) due to the switches, addressing with MAC address⁶ and IP⁷ address, and strict separation of physical media layer and protocols [Mat19]. Besides automotive-specific physical layers (cf. Section *Automotive Ethernet* on page 306), several protocols are required for real-time communication (AVB⁸ / TSN⁹, PTP¹⁰), network management, and virtual domain separation (VLAN). VLANs can separate functional domains, or different safety or cybersecurity criticality by addressing virtual sub-networks.

⁶ Media Access Control

⁷ Internet Protocol

⁸ Audio/Video Bridging

⁹ Time Sensitive Networking

¹⁰ Precision Time Protocol

Combined with the common IT protocols (IP, TCP¹¹, UDP¹², ARP¹³, DHCP¹⁴, DNS¹⁵), this causes Ethernet to be the cornerstone of a reliable and capable E/E architecture.

Remark 2.1: Network management

Network management is required to hibernate and wake up ECUs so that they require (almost) no more energy as long as the vehicle is not active. AUTOSAR¹⁶ provides protocols and a central coordinator to manage subnetworks or individual ECUs on different networks (e.g. CANNm for CAN and UDPNm for Ethernet). WoL¹⁷ is a similar IT standard.

Topologies

Originating from distributed architectures, in which few wires connect single-purpose components with little interaction (one ECU - one function), the network topologies developed to domain architectures. Here, ECUs are grouped according to their functions, and multiple buses are connected via a central gateway (cf. Figure 2.2 on page 18). Today, topologies develop to domain-centralized architectures, where multiple functions are integrated in domain controllers with higher performance, that also act as a gateway to other domains. Mercedes' STAR3 architecture, for example, connects the domain controllers via an Ethernet backbone [Sch+20a] (cf. Figure 2.3 on page 19).

Depending on the manufacturer, centralization is progressing even further. Instead of grouping by function, zonal architectures group ECUs by their location in the vehicle, which saves further wiring costs [MBB18; Bur+23]. Here, an Ethernet

¹¹ Transmission Control Protocol

¹² User Datagram Protocol

¹³ Address Resolution Protocol

¹⁴ Dynamic Host Configuration Protocol (cf. glossary, page 325)

¹⁵ Domain Name System (cf. glossary, page 325)

¹⁶ AUTomotive Open System ARchitecture

¹⁷ Wake on LAN

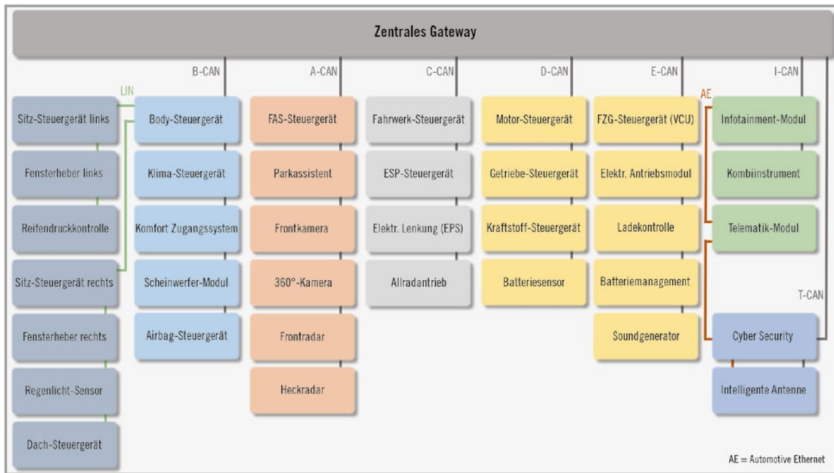


Figure 2.2: Domain architecture topology "Base2017" (©Edag) [MBB18].

backbone connects zone controllers that aggregate data locally, including legacy CAN communication. Since Tesla has been using a zonal architecture since 2017, most manufacturers are gradually following suit [Dix23]. Grouping of functions and centralization can even lead to a reduction in the number of hosts in the vehicle, contrary to previous trends. For example, Rivian was able to more than halve the number of hosts from 17 to 7 [Sha24]. Volkswagen, in comparison, moves toward full centralized computation with few strong in-car servers in its E³ architecture for the company's "ID" models [Con19] (cf. Figure 2.4 on page 19).

2.1.2 Abstraction with software architecture

To describe and structure the network communication of software systems, the OSI¹⁸ model is an established reference, that defines seven layers with distinct tasks. Within this model, software architecture can be mapped to consists of the

¹⁸ Open Systems Interconnection

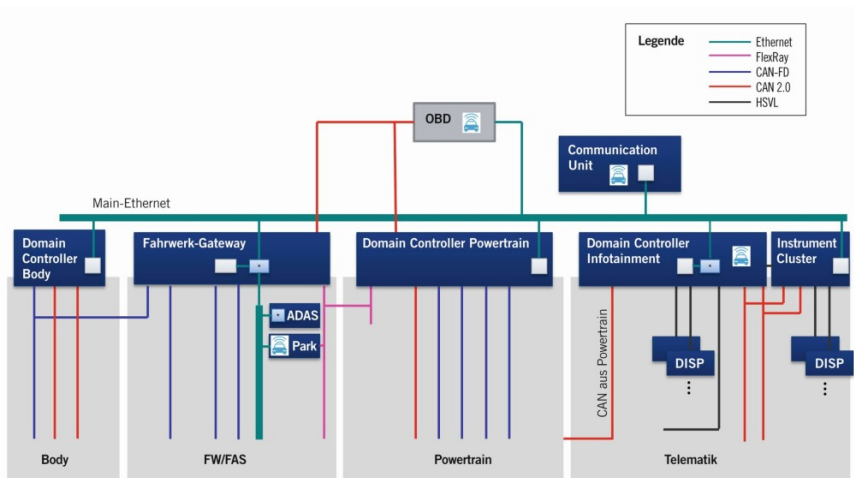


Figure 2.3: Domain centralized architecture in the STAR3 architecture (©Mercedes-Benz) [Sch+20a].

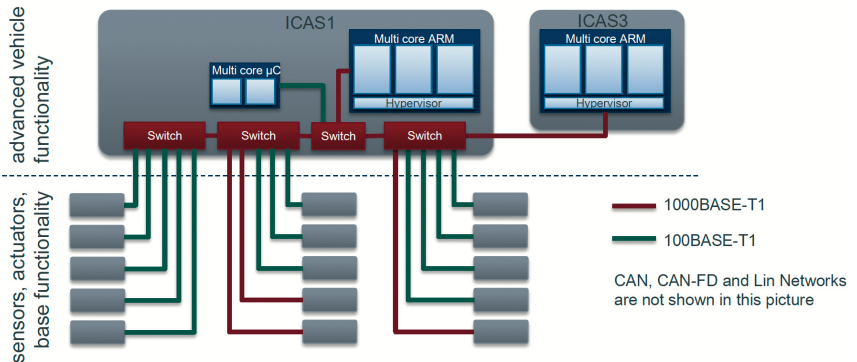


Figure 2.4: Centralized in-vehicle server architecture in the E³ architecture [hei19] (©Volkswagen / Marvell).

		AUTOSAR classic	AUTOSAR adaptive	ROS 2	
		Application	Adaptive Application	ROS 2 Node	
Operating system Middleware API	7 Application	Runtime environment (RTE)	Runtime for Adaptive (ARA)	ROS 2 client library	
	6 Presentation	SOME/IP, AUTOSAR PDU	SOME/IP, DDS, AUTOSAR PDU	ROS 2 middleware	
	5 Session			DDS	
	4 Transport	TCP, UDP		UDP	
	3 Network	IPv4, IPv6			
	2 Data Link	Ethernet, CAN			Ethernet
	Hardware abstraction	Microcontroller abstraction layer	POSIX-subset PSE51	POSIX, Windows	
	1 Physical	Hardware layer			

Figure 2.5: Schematic comparison of different software architectures in the OSI model (derived from [Sto21]).

OS on layers two to four, and the middleware¹⁹ on top in layers five to seven (cf. left side in Figure 2.5 on page 20, derived from [Sto21]). The middleware, which abstracts the network topology, yields standardized interfaces (API²⁰) on OSI layer 7, which are provided for the development of applications. The OS furthermore includes hardware abstraction between OSI layers one and two.

While historically, the E/E architecture consisted of fixed software-hardware combinations as ECUs, the SDV focuses on full abstraction of the hardware. In this sense, the SDV envisions a "fully programmable" vehicle - applications on OSI layer 7 access standardized interfaces, but mostly not specific hardware. Hence, they can be assigned to any host more or less arbitrarily.

Software architectures and development processes for embedded real-time systems in vehicles are standardized primarily by the AUTOSAR²¹ consortium [AUT03]. Currently, AUTOSAR specifies the two platforms CP²² and AP²³ [AUT17; Tis18]. Combining a real-time OS and middleware, the ACP provides

¹⁹ cf. glossary, page 327

²⁰ Application Programming Interfaces (cf. glossary, page 323)

²¹ AUTomotive Open System ARchitecture (cf. glossary, page 323)

²² AUTOSAR Classic platform (cf. glossary, page 324)

²³ AUTOSAR Adaptive platform (cf. glossary, page 324)

standardized interfaces for software on microcontrollers. However, it is not built for software flexibility, such as updating and exchanging parts. Although the ACP supports implementing a SOA using SOME/IP²⁴ over Ethernet to establish communication relationships at runtime, the ACP does not fully abstract the host, because the ECUs software is compiled as whole.

The AAP enables the effective use of multicore CPU²⁵ [Tis18] and supports SOA by the SOME/IP and DDS²⁶ protocols. Because the AAP is built on a POSIX²⁷ based operating system²⁸, it is more dynamic than the statically compiled ECUs with the ACP. Applications execute as POSIX™ processes and are loaded from RAM²⁹, which allows to change single applications on the host. Systems with the AAP can therefore build on Linux-based OS. ROS 2³⁰ is another middleware that implements a SOA, as it runs with DDS on different OS, including Linux.

Definition 2.1: Service-oriented architecture

SOA is an "architectural style that supports service orientation and is a paradigm for building business solutions." In this context, a service is a "logical representation of a set of activities that has specified outcomes, is self-contained, may be composed of other services, and is a "black box" to consumers of the service." [ISO16a].

SOA can be used to abstract signal-based communication (e.g. on CAN) in hybrid network architectures to service-oriented interfaces [Mad+23; SNI23; Sch+20a]. Dedicated services³¹, which usually run on gateways, convert the signals originally bound to specific ECUs or a bus into service interfaces in the Ethernet network. Other services can transparently access the signal-based

²⁴ Service-oriented Middleware over IP (cf. glossary, page 329)

²⁵ Central Processing Units

²⁶ Data Distribution Service (cf. glossary, page 325)

²⁷ Portable Operating System Interface (cf. glossary, page 328)

²⁸ AUTOSAR Adaptive only requires a subset of POSIX, an operating system compliant to the Minimal Realtime System Profile PSE51.

²⁹ Random Access Memory (cf. glossary, page 328)

³⁰ Robot Operating System 2 (cf. glossary, page 328)

³¹ cf. glossary, page 329

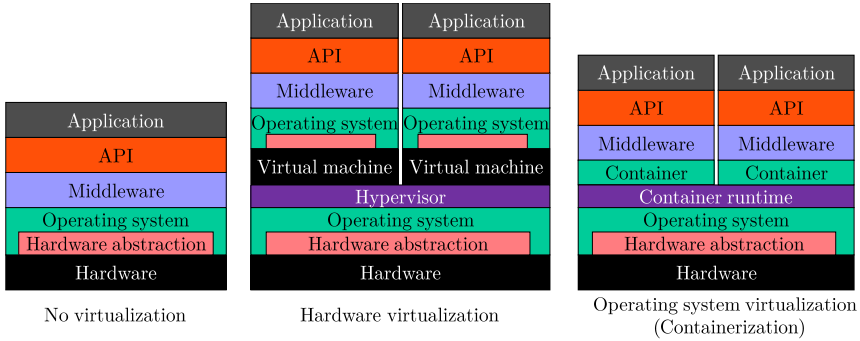


Figure 2.6: Comparison of virtualization approaches.

parts of the software architecture without having to know where the signals are implemented. Overall, a SOA-based middleware is therefore seen as an essential step for the abstraction of the hardware and the network topology [TMB17] and can enable easier integration of new features [Vet+20].

2.1.3 Virtualization

Hardware virtualization (cf. middle of Figure 2.6 on page 22) imitates a physical computer so that the code executed in a VM³² has the impression of having access to dedicated physical hardware [Sha+16]. A *hypervisor* runs on the host that provides the virtualization, which is a software that allocates hardware resources and plans the VMs’ access to them.

This allows a physical system (hardware resources like the CPU or RAM) to be split into smaller virtual parts. Since the effects of failures can be limited to a single VM, the functional safety of other VMs can be considered independently. Through these enclosed execution environments, virtualization is interesting for automotive embedded systems since several years [NDB10]. Besides safety reasons, it allows efficient use of multi-core processors by assigning software that is not inherently

³² virtual machine

parallel to individual CPU cores. Further benefits include using different OSs on one CPU, reducing network interfaces by virtual networking, and using old OS on new hardware [Str+13]. Also, software can be tested without hardware access, e.g. before hardware suppliers finish their prototypes [SNI23]. Therefore, it can accelerate the development, including verification of updates [Gui+22].

Analogous to VMs for hardware virtualization, *containerization* virtualizes the OS rather than the hardware. Containers are isolated user spaces of a shared OS kernel, giving the impression to a group of processes on the same host to operate on a dedicated OS. They simplify dependency management, because the isolation allows to bundle an application with all its software dependencies, e.g. specific versions of programming languages and third-party libraries.

Containerization consists of three aspects, standardized by the OCI³³: the container runtime, the image, and a distribution system. Images are described with plain text files, containing the commands to install dependencies, compile code, and set parameters. For many tasks, only the user's configuration needs to be added to public *base images*, which provide the required software. After a build process, that compiles and bundles the software and configuration as specified in the text file, the resulting image is downloaded and unpacked on a host using the distribution system. The runtime starts the container by isolating resources (file system, processes, network) via defined OS interfaces, and executing the commands of the unpacked image.

Containerization has no significant overhead on the reactivity of applications as long as the container's network is not isolated from the host, also on embedded devices [RSL23; KTK23]. Dedicated network adapters improve the container's cybersecurity, but communication gets slightly slower [RSL23]. While containers are more lightweight and flexible compared to VMs, one of the drawbacks is their reduced isolation [Sha+16].

³³ Open Container Initiative (cf. glossary, page 327)

2.1.4 Orchestration

Orchestration is another IT technology for software flexibility. An orchestrator controls the deployment of containerized applications to suitable hosts. Resources of the hosts, e.g. available CPU and memory, are managed by the orchestrator for scheduling, while the actual execution uses a container runtime. Besides, the orchestrator can usually control network, storage, and local interface access of containers. In IT, *Kubernetes* (k8s) is the most popular tool, which is maintained within the CNCF³⁴.

Definition 2.2: Orchestration

Orchestration is used as a "synonym of management, arrangement and coordination" [Cos+22]. It is "the composition of system components [...] to provide services to consumers" [NIS11]. The orchestrator "controls the overall workflow from starting a service until it ends with the objective to optimize and automate the [...] service deployment" [Sar+19].

In the E/E architecture, an orchestrator can distribute software components to the ECUs when the vehicle starts, but also when changes are necessary at runtime, for example, in the event of hardware failures [Sto21; Sch+22]. By automatically deploying software to hosts, an orchestrator is helpful in creating an extensible vehicle architecture [Sto+21], including the offloading of functions [Sto21; Som+24b]. Furthermore, redundancy can be achieved by running safety-relevant software components several times on different ECUs.

³⁴ Cloud Native Computing Foundation

2.2 Automotive development lifecycle

2.2.1 Lifecycle models

The automotive industry uses various lifecycle models for the development of the entire vehicle and for individual software or hardware components. These include agile as well as sequential plan-based processes such as the Waterfall model that complete a milestone with a deliverable after each development phase. Agile processes, for example Scrum, are iterative or incremental to enable a faster response to changes and promise accelerated software development and earlier discovery of problems. Key to the quality and speed of agile development are CI/CD³⁵ tools and DevOps³⁶ methodology that automate testing, release (cf. Section 2.2.3 on page 30), deployment and feedback gathering during software evolution. For example, the research project AutoDevSafeOps³⁷ based on a common DevOps lifecycle model (cf. Figure 2.7) and integrated various measures to continuously update a vehicle's software safely. When applying DevOps to cybersecurity, the term DevSecOps³⁸ is used. Here, certain cybersecurity tasks such as static code analysis and identification of software dependencies (cf. Remark 4.4 on page 86) are automated [YT22; Dup+21]. Agile development is already being used in the automotive industry for safety-critical software with several million lines of code [Ant23].

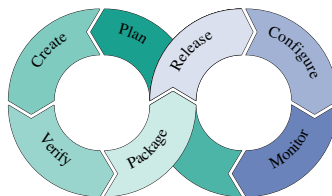


Figure 2.7: Common phases in a DevOps lifecycle model.

³⁵ Continuous integration / Continuous deployment (cf. glossary, page 325)

³⁶ Development and Operations (cf. glossary, page 325)

³⁷ cf. glossary, page 323

³⁸ Development, Security, and Operations

However, several challenges remain, such as finding the balance between documentation and safety versus flexibility [AK+24] and scaling up to an organization [Ågr+22]. Therefore, the overall product development still uses a V-model (cf. Figure 2.8 on page 28) with requirements engineering up-front in the concept phase and a strict testing and validation phase before moving to production. In particular, the ISO³⁹ 26262 [ISO18b] standard for functional safety (cf. Section *Functional safety* on page 34) in the automotive domain follows a V-model.

The requirements analysis and system design are on the left-hand side. Moving down, based on the technical requirements, the system architecture and the interfaces, software and hardware requirements are established. Here, the system development splits up into hardware and software development. On the software side, first the software architecture is defined and afterward the software is designed in detail, and implemented. The integration steps on the right-hand side of the V-model match the corresponding steps on the left-hand side by verification against the requirements and design (cf. arrows in Figure 2.8 on page 28).

Traditionally, vehicle development is broken down into subsystems [Sta21] (cf. Figure 2.8 on page 28) along a hierarchical structure of functions, which are assigned to a domain. The functions a customer can observe are called features (cf. Definition 2.4 on page 27). In distributed development, the meaning of abstraction levels (system, subsystem, etc.) may differ. For example, while for an OEM customer the subsystem level can be a specific hardware component, such as an ECU, or a software component, from an ECU supplier's point of view (Tier 1), the ECU, respectively the software component, is the entire system. In this thesis, a system generally denotes the vehicle as a whole. Components are the elements realizing the functional characteristics, both the ones visible to the end-user (features), as well as non-visible functions.

³⁹ International Organization for Standardization

Definition 2.3: System

A system is defined as the "combination of interacting elements organized to achieve one or more stated purposes" (ISO/IEC/IEEE 24765:2017 "Systems and software engineering — Vocabulary" [ISO17]).

Definition 2.4: Feature

A feature is an "abstract functional characteristic of a system of interest that end-users and other stakeholders can understand" (ISO/IEC 26550:2015 [ISO15c]).

Definition 2.5: Component

A component is a "functionally or logically distinct part of a system. [...] A component can be hardware or software and can be subdivided into other components" (cf. ISO/IEC/IEEE 24765:2017 [ISO17]).

2.2.2 Model-based systems engineering

To cope with the complexity of vehicles, which manifests in more than 45,000 in-vehicle network signals, thousands of functions, and millions of lines of code [VS21; Gui+18; Ant20], the automotive industry adopted MBSE. It breaks down the complexity by formalizing the description of the system and its components [Bau+23; Sta21]. Different views for the development of the system architecture are defined according to the steps of the development process. One or more models are created for each view, for example using UML⁴⁰, SysML⁴¹, or specialized tools such as PREEvision [Vec24].

⁴⁰ Unified Modeling Language

⁴¹ Systems Modeling Language

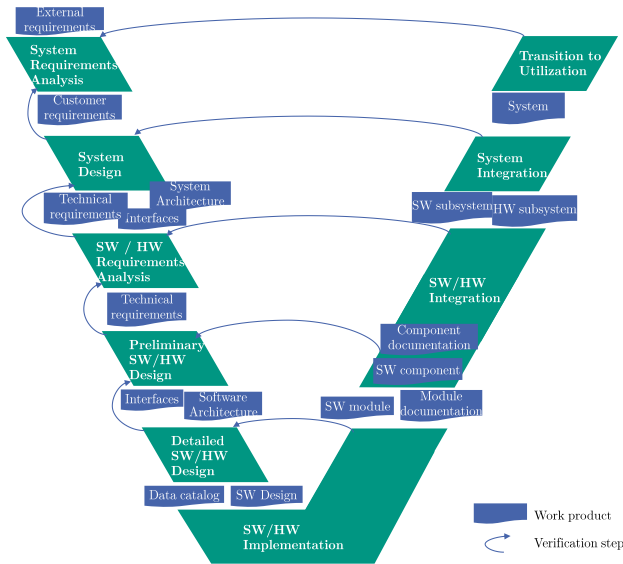


Figure 2.8: The system development sub-model of the V-model '97 (derived from [DW99]), originally developed for the IT systems of the Federal Republic of Germany. "Subsystem" replaces the original term "unit".

Definition 2.6: View

A view is defined as a "work product expressing the architecture of a system from the perspective of specific system concerns" (from ISO/IEC/IEEE 42010:2011 [ISO11]).

A popular approach for view-based development is the RFLP⁴² methodology [Poh+12]. *Physical* is also referred to as *Technical* view. Several views and work products may be associated with each step in the V-model in RFLP, e.g. R and F-view for the system requirement analysis [Poh+16] could involve a use case diagram (R) and a plain function list (F). The views can be re-applied at a finer granularity for deeper development steps. In practice, the separation between functional and logical view is a challenge, because both are a decomposition of

⁴² Requirements - Functional - Logical - Physical (cf. glossary, page 328)

the system. For example, PREEvision even has a layer called “logical functional architecture,” which can be seen as a combination of both views. This work differentiates as follows:

Definition 2.7: Functional architecture

The functional architecture describes the system from the black-box perspective by expressing all technical functions required to implement the features of the system, including their interrelations and dependencies (derived from [KŞV+22] and [Bro+09]).

Definition 2.8: Logical architecture

"The logical architecture describes the [...] system from the white-box-perspective. It reflects the decomposition of a system into a number of logical components that interact and cooperate to offer the functionality, described in the functional architecture" (from [Bro+09]).

Definition 2.9: Physical architecture

The physical / technical architecture describes the system from the realization perspective by modeling the implementation of logical components as software and hardware. It consists of the software, hardware, networks, and the allocation of software to hardware (derived from [Bro+09]). Within this work, the term E/E architecture refers to the technical view (cf. Figure 1.1 on page 2).

The technical view can include, for example, models of software architecture and behavior, network topology, hardware component architecture (e.g. micro-controllers in an ECU), and geometrical and electrical modeling of the wiring harness. Based on these definitions and the development model, an RFLP-based development could proceed as follows: First, requirements analysts define the requirements (R) and based upon that, high-level features (cf. Definition 2.4 on page 27) to fulfill the customer's demands (F). The system architects refine the features to a functional architecture (F) and define a logical architecture (L) during

system design phase. The definition of the E/E architecture (P) is the task of the software and hardware architects, in cooperation with the system architects, based on the defined logical system architecture. Software architects define the software components and their sub-components, while the hardware architects define the network topology, hardware and, if applicable, the assignment of software to hardware (P). The implementation of the designed software and hardware components (and sub-components) is the task of the software and hardware developers (P).

2.2.3 Variant and release management

Definition 2.10: Variant

A variant is "one alternative that may be used to realize particular variation points" (cf. ISO/IEC 26550:2015 [ISO15c]). In this thesis, a variant means a viable selection of features realizing all variation points.

The variation points are the variable characteristics of the managed products, such as different code for different selections of automated driving features. Customers configure vehicles individually, and each variant offered must be feasible and safe. Variant management therefore accounts for offering safe products, based on maintaining and selecting viable software and hardware realizations for different customer demands. Due to the many variants in which vehicle models are to be built, this discipline is becoming increasingly relevant [Gui+18]. Variant management usually models constraints between features, such as feature *A* is required for feature *B* (AND-constraint). Other common constraints are OR (both *A* and *B* can be selected), and XOR (only one of *A* or *B*). Feature models, which are seen as part of the functional architecture, are an established means to formalize the variation points, such as in PREEvision and AUTOSAR [AUT21]. On top of modeling features, often a SPLE⁴³ approach is used [Kni+22; Vec24].

⁴³ Software Product Line Engineering

Definition 2.11: Product line

A product line or product family is a "set of products and/or services sharing explicitly defined and managed common and variable features and relying on the same domain architecture to meet the common and variable needs of specific markets" (ISO/IEC 26550:2015 [ISO15c])

Remark 2.2: 150 / 100% E/E architecture models

Product lines in automotive MBSE are usually formalized as a 150% model, that contains all available logical and technical architecture elements of a vehicle series, including mutually-exclusive concepts. From a 150% model, the 100% model of a selected variant can be derived.

In the automotive sector, development projects are usually based on a stage gate model [Coo08] that bundles activities into phases (stages) and concludes and synchronizes them with milestones (gates). Further development can be discontinued after each stage if the risks are too high and the milestone is not reached. The gates are usually related to hardware samples (A/B/C/D samples) with increasing maturity [Sax08]. Software changes are bundled in releases, whereby a release plan specifies which functions are implemented at a particular time. In particular, releases are created when reaching a gate with a hardware sample. Release management includes the processes and metrics to authorize the changes.

Definition 2.12: Release

A release is a "particular version of a configuration item that is made available for a specific purpose." (IEEE 828-2012 [IEE12])

Definition 2.13: Version

A version is an initial release or re-release of a configuration item, identified by a unique string of number and letter values indicating a unique revision of the item. (derived from IEEE 828-2012 [IEE12] and ISO/IEC 19770-5:2015 [ISO15a])

This work extends the term *version* to hardware configuration items, which refers to completed further developments of a hardware component, as opposed to A-to-D prototype stages. In general, a *release* refers to a specific feature or the vehicle's software under configuration management as a whole. A particular release of the whole product line is also termed *system revision* in this work [Ana+20].

2.2.4 Update development and deployment

Besides managing variants and releases during development, software updates after production require managing different software versions over the vehicle's lifecycle. As software updates involve creating a release for a software component or the whole vehicle, this work considers update development as a task of variant and release management.

The V-model (cf. Figure 2.8 on page 28) does not include phases beyond the completion of the development process. In the past, software was further developed mainly to address problems that arose during customer usage. For safety-critical issues, affected vehicles were recalled to workshops where updates were installed locally via diagnostic interfaces (e.g., OBD). With OTA updates, the update delivery process became faster. While the scope of OTA update-ability varies between the manufacturers, several OEMs are ready to deliver new software to most of the vehicle's hardware components [Fis23]. Still, not all vehicles install updates as soon as they become available, as the customers may reject or postpone the update (Example: Tesla⁴⁴).

⁴⁴ <https://teslafi.com/firmware>

Since the development of updates must also comply with the relevant standards and regulations, dedicated development lifecycle models for updates are being researched, e.g. V-model based or agile [Sch+20b; Gui+22; Dob+23]. The standard ISO 24089 [ISO23b] defines basic procedures for developing safe updates and combining them to *software update packages*, that consist of the software and associated data. The main legal basis is UN Regulation 156 "Software update and software update management system" [EEC21b]. The SUMS⁴⁵ includes both technologies in the vehicle and processes and shall ensure that software that has been approved (homologated) remains safe after an update. If updates fail, a safe state must be reached, e.g. restoring the system to the previous version (rollback) or deactivating functions.

Technically, the update capability must be supported by software (distribute and exchange code) and hardware (enough resources) of the vehicle. A centralized network topology can be helpful, as it limits the number of control units to be updated, which can simplify update schedules. Virtualization (cf. Section 2.1.3 on page 22) also strengthens the update capability, for example, to safely roll out new versions of software in parallel to the one currently running [GZS23].

AUTOSAR standardizes OTA updates through the Update and Configuration Management (UCM) master and clients within the AAP. This allows manufacturers to leverage a consistent approach across different hosts. Cybersecurity is ensured by encryption and signatures, and rollbacks guarantee reliability, if necessary [AUT23d; AUT23e]. While the ACP can support software updates, its capabilities are limited, because a full re-flash of the ECU's software is required. Thus, the AUTOSAR Foundation speaks of FOTA⁴⁶, not of dedicated software updates. However, ACP ECUs may be updated via the diagnostic protocol UDS, which is used by workshops via external diagnostic tools communicating with a software component in the vehicle [AUT23f]. Focusing on future E/E architectures, this dissertation assumes full OTA update capability for both microcontroller-based as well as high-performance hosts.

⁴⁵ software update management system

⁴⁶ Firmware-over-the-air

Remark 2.3: Update management service

This thesis refers to an in-vehicle software component that provides OTA update capability for both microcontroller-based and POSIX-based hosts using the OSs and middleware as *Update management service*.

2.2.5 Safety

Functional safety

There are several similarities and parallels between the development of safe and secure vehicles. In the context of this work, “safety” refers to functional safety as defined by ISO 26262 [ISO18b], which has been established in the automotive industry for several years. While functional safety aims to protect the vehicle’s surroundings (passengers, the environment, other road users) from the vehicle, the goal of cybersecurity is to protect the vehicle from its surroundings (attackers). However, ISO 26262 focuses on faults in E/E systems (e.g. timing violations, corrupted messages) to ensure everything works as expected, but not hazards arising from system limitations *if everything works as expected*, which is discussed under the term SOTIF⁴⁷ (cf. Section *SOTIF and Operational design domain* on page 37).

Definition 2.14: Functional safety

While ISO 26262 generally defines safety as the "absence of unreasonable risk", in the context of vehicles the term *functional safety* defines the "absence of unreasonable risk due to hazards caused by malfunctioning behaviour of E/E systems" [ISO18b]. In other words, "there shall not be any harm to persons resulting from faults in electronics or software" [Sta21].

⁴⁷ Safety Of The Intended Functionality

Definition 2.15: Risk

In the context of safety, *risks* are defined as the "combination of the probability of occurrence of harm and the severity of that harm." [ISO18b]

Definition 2.16: Hazard

A *hazard* is a "potential source of harm caused by malfunctioning behaviour of the item." When tied to the situation in which occurs, a *hazardous event* is the "combination of a hazard and an operational situation." [ISO18b]

The definitions of cybersecurity (cf. Definition 1.1 on page 3, Remark 2.4) and safety (cf. Definition 2.14, Definition 2.15) share both the task of weighing and evaluating risks ("absence of unreasonable risk" and "sufficiently protected," respectively) and the inherent uncertainty in risk assessment ("probability of occurrence" and "effect of uncertainty," respectively). This gives rise to the analogies between hazard and threat (cf. Definition 2.20, Definition 2.16), and accordingly also to the objectives of HARA⁴⁸ and : the identification and assessment of risks, and, building on this, defining how to reduce the risks. Similarly to the TARA (cf. Section *Overview of the TARA steps* on page 42), the HARA is applied in the concept phase to specify the according requirements.

Specifically, in the HARA [ISO18c], the *item* of the analysis is first defined; this is typically a vehicle component that implements a vehicle function at the system level. The *items* for the TARA are exactly the same in ISO/SAE 21434. ISO 26262 then prescribes a systematic identification of hazards and the situations and system modes in which malfunctions of the item are relevant, in order to define all relevant hazardous events.

Unstructured methods such as brainstorming may be used in hazard identification solely to support the recommended structured methods such as FMEA⁴⁹ or

⁴⁸ Hazard Analysis and Risk Assessment

⁴⁹ Failure Mode and Effects Analysis

HAZOP⁵⁰. The most prominent method is FMEA, which employs a multi-step analysis that progresses from the system structure through system functions to the malfunctions. However, current research is also exploring other methods that are becoming increasingly important, particularly for automated vehicles [SLZ21].

Each identified hazardous event is classified based on its risk to determine an ASIL⁵¹ (ASIL A through ASIL D). Events with very low risk are covered by quality management (QM), while the highest risks pose the highest integrity requirements, classified as ASIL D. The actual risk of a hazard depends on the frequency of the hazardous event, the controllability (C), and the severity (S) of the harm. The frequency of the event, in turn, depends on the frequency of faults and the frequency of the relevant operating conditions. In risk assessment, ISO 26262 simplifies hazardous event frequency to exposure (E), which is the probability of being in a relevant operating situation. The ASIL is thus determined using a look-up table with discrete ordinal values for *S* (0–3), *E* (0–4), and *C* (0–3).

Safety goals are defined in accordance with ASIL, from which a *functional safety concept* is developed in the form of *safety requirements*. The safety goals are high-level requirements on the item (what to do or prevent), while the functional safety concept specifies how to achieve that, such as requiring a fault-tolerant time interval or being able to reach a safe state. Compliance with defined safety requirements must not only be tested and verified during development, but is also typically monitored during operation. This is achieved, for example, through watchdogs that monitor execution times and intervals, or error-detecting and correcting codes for memory and messages. If errors are detected, the vehicle can switch to a safe but functionally reduced *limp mode*. This mode has also been used by researchers for security purposes (cf. Section *Responses in vehicles* on page 92).

Safety analysis also plays a role in cybersecurity analysis. Given the similarities in methodology described above, there are ongoing discussions about how HARA

⁵⁰ Hazard and Operability Study

⁵¹ Automotive Safety Integrity Level (cf. glossary, page 323)

and TARA can be synchronized or systematically conducted together [För+19; Lie21; TSS22]. These include, for example, the approaches that a safety-relevant item (identified ASIL A–D) is in any case security-relevant, and that safety goals should be included as assets in the TARA. Threats with safety implications should already have been identified in the HARA; their impact rating is determined by the ASIL classification (cf. Section c) on page 45). Furthermore, there are challenges in reconciling cybersecurity and safety, as a cybersecurity mechanism can certainly have negative or positive effects on safety, and vice versa [Gra+19; PST22]. For example, message plausibility checks for safety might improve security [Gra+19], while V2X⁵² message encryption improves security, but increases latency, which reduces safety of depending features [PST22].

SOTIF and Operational design domain

The standard SAE⁵³ J3016 [SAE18] classifies the automation of vehicles into six levels, spanning from no automation (Level 0) over today’s widely available partial automation capabilities of Level 1 and 2 (e.g. cruise control, lane keep assistance, emergency braking) to full automation, i.e. driving in all situations without intervention of the passengers.

As automation increases (particularly at Levels 3, 4, or 5), the system’s responsibility for safety continues to grow. Even if everything has been done correctly in safety engineering and the software and hardware are functioning flawlessly, this does not necessarily mean that the vehicle will behave safely: the intended behavior may still be unsafe and, in the event of incorrect environmental perception and erroneous conclusions, lead to hazards. This is why the term SOTIF is gaining importance [ISO22].

⁵² Vehicle to Everything

⁵³ Society of Automotive Engineers

Definition 2.17: Safety Of The Intended Functionality

The SOTIF (Safety Of The Intended Functionality) is defined as "the absence of unreasonable risk due to hazards resulting from functional insufficiencies of the intended functionality or by reasonably foreseeable misuse by persons" [ISO22].

One key aspect toward SOTIF is to define the capabilities and limitations of an ADS⁵⁴ formalized in a specification of the ODD⁵⁵ and ODC⁵⁶. Originating from the testing perspective, ISO 34503 and ISO 34504 define the term ODD [ISO23a; ISO24b] and a taxonomy of attributes for specifying an ODD. While the ODD covers all specified operating conditions, a specific situation currently prevailing during operation is described by the term COD⁵⁷.

Definition 2.18: Operational Design Domain and Conditions

The ODD (Operational Design Domain) "represents the operating conditions within which an ADS can perform the dynamic driving task safely during a trip" [ISO23a]. The ODC represent all "potentially necessary attributes for the overall design of an ADS", including the ODD, vehicle status and behavior, driver and passenger status, and other necessary conditions, e.g. behavior of other road users [ISO23a].

In addition to the HARA, the SOTIF process aims to identify and eliminate functional insufficiencies during development. The space of situations in which remaining insufficiencies may cause hazardous behavior is to be reduced accordingly. The SOTIF process therefore requires the identification of potential *triggering conditions*, which are operating conditions (within or outside the defined

⁵⁴ Automated Driving System

⁵⁵ Operational Design Domain

⁵⁶ Operational Design Conditions

⁵⁷ Current Operational Domain

ODD) that could lead to potentially unsafe behavior. All hazardous events identified as a consequence must either be eliminated through (modified) design or managed through operational monitoring to ensure safe behavior.

Definition 2.19: Current Operational Domain

The COD (Current Operational Domain) is "the specific set of operating conditions which exists presently in the immediate vicinity of an ADS, including, but not limited to, environmental, geographical, and time-of-day restrictions, and/or the requisite presence or absence of certain traffic or roadway characteristics." [ISO23a]

The integration of ODCs into the development lifecycle, including design, testing and operation, is still the subject of research. Technically, a report of ISO [ISO20b] recommends to implement an *ADS mode manager*, that should check whether the attributes of the ODD and ODC of a given ADS are fulfilled, i.e. if the COD are within the ODC. If not, the automated driving functionality will be deactivated, or a MRM⁵⁸ is initiated to maintain a reasonable safe state even within degraded operating conditions. In addition, continuous monitoring of the vehicle is one of the aspects currently under discussion to develop improvements to the system, increase the ODD, and identify and address potential shortcomings [NBT22; Pia22] (see also Section *Physical situation monitoring* on page 79).

2.3 ISO/SAE 21434 Cybersecurity Engineering

In 2021, the harmonized standard ISO/SAE 21434 Cybersecurity Engineering came into force, that overrides and renews SAE J3061 [SAE16], which was the first standard for the cybersecurity of vehicles. ISO/SAE 21434 defines a cybersecurity engineering framework that covers all aspects of the automotive lifecycle. Fig. 2.9 depicts all the processes defined in the standard and places them in the corresponding lifecycle phase of the V-model (cf. Section 2.2 on page 25,

⁵⁸ minimal risk maneuver

Figure 2.8 on page 28). The position of the colored blocks shows which development lifecycle phase is targeted by the corresponding clauses in the standard, while the document shapes depict the work products⁵⁹. Clauses 5, 6, and 7 describe overarching requirements, namely organizational and project dependent cybersecurity management (5 and 6), and distributed engineering activities (7). The main outcome of clause 7 is the cybersecurity interface agreement, which specifies the responsibilities of customer and supplier. The development phase is covered by clauses 9 and 10 and concluded with tests and integration (clause 11) (cf. Section 2.3.1). In comparison to the V-model, additional phases after SOP⁶⁰ are considered, namely production, operations and maintenance, and decommissioning. ISO/SAE 21434 denotes clauses 12, 13, and 14 as *post-development phases* of the cybersecurity engineering lifecycle. In addition, clauses 5 and 8 are relevant for cybersecurity operations (cf. Section 2.3.2).

2.3.1 Development processes

Following the "V", clause 9 covers the definition of a cybersecurity concept, containing the cybersecurity requirements and requirements on the operational environment (cf. Figure 2.10 on page 42). The focus of requirements elicitation are functions at vehicle level, implemented by *items*. The TARA methodology of ISO/SAE 21434 (clause 15) is applied to derive high-level requirements (*cybersecurity goal*), that aim to protect assets against threat scenarios. Relevant terms include the *item*, *attack path*, *threat scenario* and *damage scenario* (cf. Figure 2.11 on page 42 and Figure A.1 on page 273), which will be discussed in the following (cf. Section *Overview of the TARA steps* on page 42). Generally, the TARA is not seen to be part of the concept phase, but could be invoked from any point in time in the lifecycle. However, TARA is mandatory in the concept phase to ensure adequate efforts for identifying and prioritizing cybersecurity goals. ISO/SAE 21434 does not mandate a specific overarching method for TARA, but

⁵⁹ "Cybersecurity" is shortened to "CS" and some work products are combined to one shape

⁶⁰ Start of production

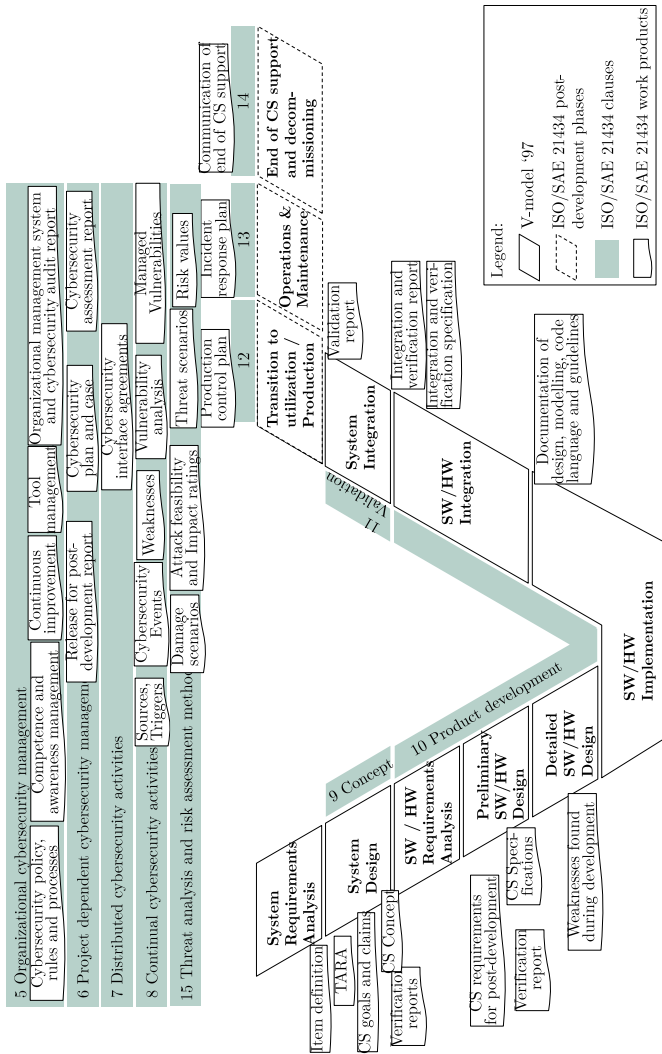


Figure 2.9: Cybersecurity engineering according to ISO/SAE 21434 and alignment of the work products and clauses in the V-model.

recommends several approaches for its sub-steps. This thesis mainly follows the HEAVENS 2.0 approach, which covers all necessary TARA steps and is therefore considered a suitable overarching approach (cf. Section *Example: HEAVENS 2.0 workflow* on page 47).

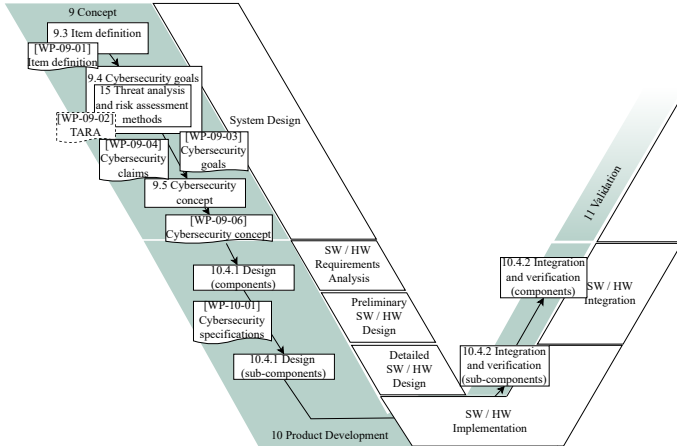


Figure 2.10: The concept and product development phases (clause 9 and 10) in detail.

Overview of the TARA steps

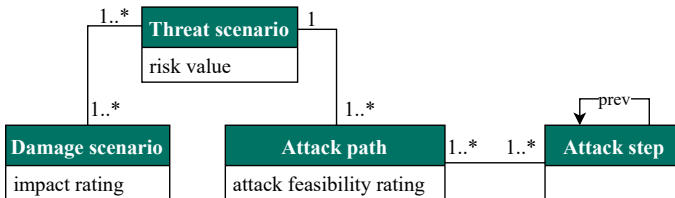


Figure 2.11: TARA outcome: artifacts and their multiplicity (derived from [LAO21], extended with attack step).

a) Threat scenario identification The *threat scenario* (cf. Definition 2.20) is the term used in ISO/SAE 21434 to define a theoretical attack and its risks. The overall goal of TARA is to identify all threat scenarios and assign a *risk value* (cf. Remark 2.4) to each of them, based on an assessment of the impact of the associated damage scenario(s) and the and feasibility of the respective attack path(s).

Definition 2.20: Threat scenario

Threat scenario means a "potential cause of compromise of cybersecurity properties of one or more assets in order to realize a damage scenario" as of ISO/SAE 21434 [ISO21].

Definition 2.21: Cybersecurity property

A *cybersecurity property* is an "Attribute that can be worth protecting." based on ISO/SAE 21434 [ISO21]. Note: Cybersecurity properties commonly include availability, integrity, and confidentiality, but may also include non-repudiation, authentication and authorization.

Remark 2.4: Risk value

A cybersecurity risk is the "effect of uncertainty on road vehicle cybersecurity expressed in terms of attack feasibility and impact." [ISO21] ISO/SAE 21434 expresses particular cybersecurity risks as a value between (and including) 1 and 5, where 1 represents minimal risk. Threat scenarios with a risk value of 1 are usually not considered for further analysis and treatment.

ISO/SAE 21434 elaborates that threat scenarios could be identified in group discussions, or systematic approaches like misuse-cases and threat modeling. Common threat modeling methods can be applied to a system model to create a threat model, which can be textual (tables, lists) or graphical (e.g., attack trees

and attack graphs). Microsoft's STRIDE⁶¹ model [KG99; Sho14], for example, is a mnemonic of six main categories of software threats (cf. Table 2.1). Applied to a DFD⁶², the method yields a list of possible threat scenarios and assets of a system.

Table 2.1: STRIDE model (from [Sho14]).

Threat	Desired property	Threat Definition
Spoofing	Authenticity	Pretending to be something or someone other than yourself
Tampering	Integrity	Modifying something on disk, on a network, or in memory
Repudiation	Non-repudiability	Claiming that you didn't do something or were not responsible; Repudiation can be honest or false
Information disclosure	Confidentiality	Providing information to someone not authorized to see it
Denial of service	Availability	Absorbing resources needed to provide service
Elevation of privilege	Authorization	Allowing someone to do something they're not authorized to do

b) Attack path analysis and feasibility rating One or more *attack paths* (cf. Definition 2.22) realize a specific threat scenario (cf. Figure 2.11). A threat scenario that cannot be realized (no attack path known) is not relevant. The categories of attack feasibility are predefined in the standard (cf. Definition 2.22), while various methods can be used to determine them.

Definition 2.22: Attack path and attack feasibility

An *attack path* is defined as "set of deliberate actions to realize a threat scenario" in ISO/SAE 21434. The *attack feasibility* describes "the ease of successfully carrying out the corresponding set of actions." The attack feasibility rating of each attack path ranges from *very low* ("The attack path can be accomplished utilizing very high effort") over *low* and *medium* to *high* ("The attack path can be accomplished utilizing low effort.") [ISO21].

⁶¹ Spoofing - Tampering - Repudiation - Information disclosure - Denial of service - Elevation of privilege

⁶² Data flow diagram

For the attack path analysis, ISO/SAE 21434 proposes bottom-up (from identified vulnerabilities) or top-down approaches using threat models such as attack trees or attack graphs. For the feasibility rating, annex G outlines guidelines for three different approaches, namely an attack potential-based, a CVSS⁶³-based, and an attack vector-based approach. They incorporate different aspects and calculations, such as window of opportunity and knowledge of the item (attack potential), or required privileges and preconditions (CVSS). An attack potential-based approach is used, for example, in the HEAVENS 2.0 workflow (cf. 47).

An attack path is usually not just a single action (cf. Definition 2.22). Even the example in Annex H of ISO/SAE 21434 describes attack paths consisting of four steps that (must) take place one after the other to realize the threat scenario. Some steps may be interchanged, but must be completed successfully before the next step can be executed.⁶⁴ Accordingly, this thesis formalizes the attack step as an additional artifact resulting from TARA, that can be part of multiple attack paths, while an attack path consists of a number of attack steps (cf. Figure 2.11)

Definition 2.23: Attack step

An attack step as a specific element of an attack path, which can be assigned a particular position in the order of realization of the attack path.

c) Impact rating of damage scenarios The realization of a threat scenario causes at least one *damage scenario* (negative consequence, cf. Definition 2.24), whereby a particular consequence can also be a result of different threat scenarios. The standard requires an assessment of the impact (cf. Definition 2.25) on the road user in terms of safety (no injuries to life threatening), operational (non-perceivable impairment of function to complete loss), financial (negligible to catastrophic) and privacy (negligible to irreversible).

⁶³ Common Vulnerability Scoring System (cf. glossary, page 324)

⁶⁴ Example: In principle, it does not matter whether the user name or the password is found out first if the next attack step is to log in with legitimate user data. However, both pieces of information are required.

Definition 2.24: Damage scenario

ISO/SAE 21434 defines the *damage scenario* as "adverse consequence involving a vehicle or vehicle function and affecting a road user" [ISO21].

Definition 2.25: Impact rating

The *impact* is defined as an "estimate of magnitude of damage or physical harm from a damage scenario". To assess the *impact rating* of a damage scenario, the consequences regarding safety, privacy, finances and operation should be taken into account [ISO21]. The impact rating of each of these categories is either *severe*, *major*, *moderate*, or *negligible*.

Overall, the outcome of the TARA includes three types of artifacts for threat scenario, damage scenario and attack path with the corresponding relationships (cf. Figure 2.11 on page 42) and ratings. Upon the TARA artifacts, particularly upon the risk value, decisions for or against the implementation of countermeasures are taken to reduce or eliminate the risk of the particular threat scenarios. Instead of reducing the risk by introducing cybersecurity controls, the manufacturer could avoid the risk by changing the architecture, sharing the risk through contracts with suppliers or insurance, or accepting the risk as it is. Acceptance and sharing yields the *cybersecurity claims*, while reducing the risk yields *cybersecurity goals*. The final step of clause 9 is to define the *cybersecurity concept*, consisting of cybersecurity requirements that are derived (refined) from the goals (cf. Figure 2.10). While clause 9 corresponds to the system-level design, clause 10 covers the component-level design and development by creating the (detailed) cybersecurity specifications, documentation and reports for integration, and verification. On system level, clause 11 covers validation reports, among other work products.

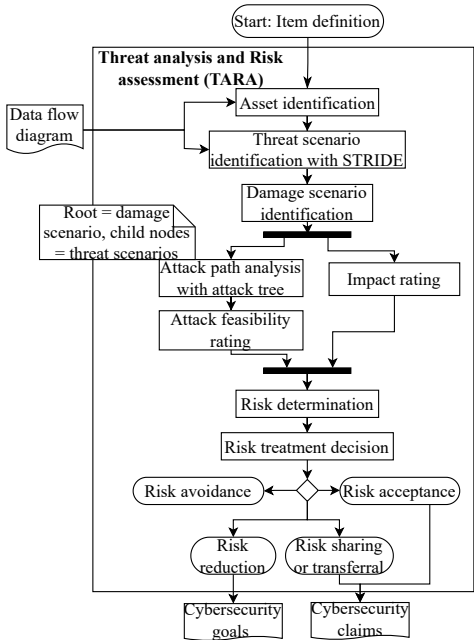


Figure 2.12: The HEAVENS 2.0 TARA methodology, based on [LAO21]

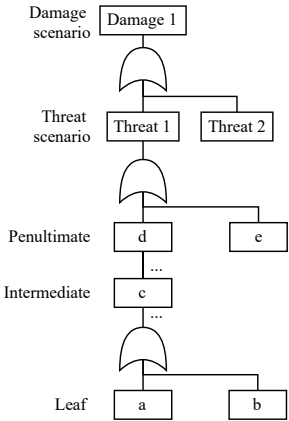


Figure 2.13: Nomenclature for the position of attack steps in attack trees.

Example: HEAVENS 2.0 workflow

An ISO/SAE 21434-compliant workflow is the HEAVENS 2.0 model (cf. Figure 2.12) [LAO21], for example, which updates its established predecessor [LI16] to adapt it to industry experience. It uses attack trees to model attack paths, and STRIDE on a DFD to derive threat scenarios. In attack trees as used by HEAVENS 2.0, *damage scenarios* are the root nodes (uppermost), below follow the *threat scenarios* (cf. Figure 2.13). At the bottom of the tree are the initial attack vectors, the *Leaf* nodes. To differentiate the attack steps between threat scenario and leaf, this thesis introduces the nomenclature (cf. Figure 2.13) *Penultimate* (last step before the realization of the threat scenario) and *Intermediate* for all further steps. An attack tree according to HEAVENS 2.0 is usually at least three steps

deep (damage scenario, threat scenario, leaf). Further attack steps (penultimate, arbitrary number of intermediate) can be added.

HEAVENS 2.0 applies an attack potential based method for feasibility rating. The overall attack feasibility is calculated by averaging ($AF = \frac{\sum_i af_i}{3 \cdot n}$) over the estimated parameters af (cf. Table 2.2) and classification of the numerical value into four classes (cf. Table 2.3). To assess the impact, the categories Safety, Operational, Financial and Privacy (of the road user) are considered as required by ISO/SAE 21434. HEAVENS evaluates the impacts (im) in these criteria from None (0) to High (100) and arrives at the impact rating through averaging ($IM = \frac{\sum_i im_i}{100 \cdot n}$) and classification (cf. Table 2.4).

Table 2.2: Parameters af of attack feasibility, cf. Table 5 in [LAO21].

Expertise	Value	Knowledge of item	Value	Window of opportunity	Value	Equipment	Value
Multiple Experts	0	Critical	0	Small	0	Multiple bespoke	0
Expert	1	Sensitive	1	Medium	1	Bespoke	1
Proficient	2	Restricted	2	Large	2	Specialized	2
Layman	3	Public	3	Unlimited	3	Standard	3

Table 2.3: Calculation of attack feasibility rating, **Table 2.4:** Calculation of impact rating, cf. Table 6 in [LAO21]

Parameter sum (AF)	Attack feasibility rating
$0.00 \leq x < 0.30$	Very Low
$0.30 \leq x < 0.60$	Low
$0.60 \leq x < 0.80$	Medium
$0.80 \leq x \leq 1.00$	High

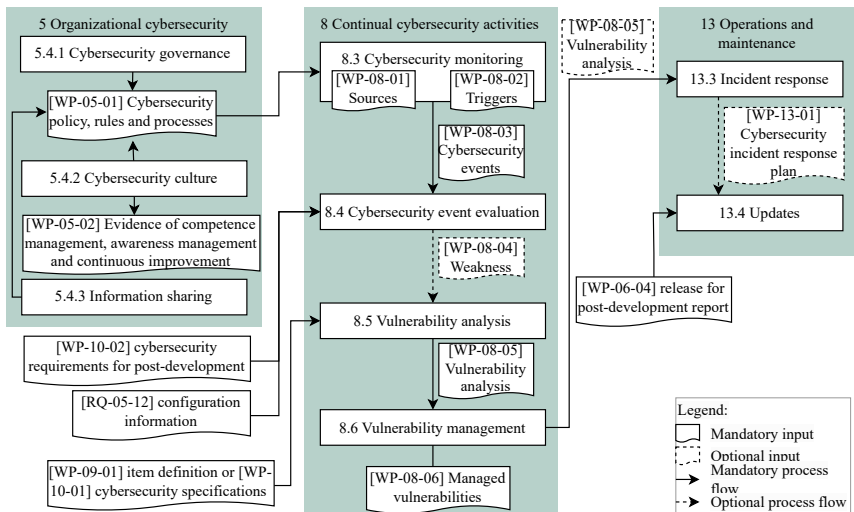
Parameter sum (IM)	Impact rating
$0.00 \leq x < 0.01$	Negligible
$0.01 \leq x < 0.05$	Moderate
$0.05 \leq x < 0.45$	Major
$0.45 \leq x \leq 1.00$	Severe

2.3.2 Processes for cybersecurity operations

Cybersecurity activities after development cover in ISO/SAE 21434 production (clause 12), operations and maintenance (clause 13), and the end of cybersecurity support when the vehicle is decommissioned (clause 14). In this work,

Table 2.5: Matrix to calculate risk value, cf. Table 8 in [LAO21].

		Impact rating			
		Negl.	Mod.	Maj.	Sev.
Attack feasibility rating	Very Low	1	1	2	3
	Low	1	2	3	4
	Medium	2	3	4	5
	High	2	4	5	5

**Figure 2.14:** Processes covering continuous and post-development activities in ISO/SAE 21434 (overview: cf. Figure 2.9 on page 41)

however, the focus of the post-development processes is placed on cybersecurity operations, where in addition to clause 13, the clauses 5 and 8 are relevant (cf. Figure 2.14 on page 49). The continual cybersecurity activities of clause 8 are not project-specific: Since newly discovered weaknesses and vulnerabilities may also influence ongoing development projects, clause 8 covers the whole development lifecycle (cf. Figure 2.9 on page 41), not only post-development. clause 5 (organizational cybersecurity management) has a direct influence on the operations phase by providing several mandatory inputs for clause 8.

Definition 2.26: Cybersecurity information

Information with regard to cybersecurity for which relevance is not yet determined.

Definition 2.27: Weakness

Defect or characteristic that can lead to undesirable behavior

Definition 2.28: Vulnerability

Weakness that can be exploited as part of an attack path.

Clause 5 defines the required processes and rules for the subsequent activities, including clauses 8 and 13, for information sharing, and a continuous improvement process, that should cover all cybersecurity activities. Besides, retaining configuration information until the end of cybersecurity support is mandatory (cf. [RQ-05-12], [ISO21]).

Cybersecurity monitoring

In cybersecurity monitoring (clause 8.3), events are generated based on selected information sources ([WP-08-01]) and triggers (e.g., keywords in documents, [WP-08-02]). While ISO/SAE 21434 calls triage to be the process step of filtering information by triggers, this work refines the term (cf. Definition 7.5 on page 137).

To select information sources ([WP-08-01]), ISO/SAE 21434 recommends considering vehicle manufacturer-internal and external sources. Internal ones include development artifacts (cybersecurity specifications, TARA artifacts), past vulnerability analysis, and information from the field, such as vulnerability scanning reports (cf. Remark 4.3 on page 85), repair information, and consumer usage information. External sources include researchers, commercial and non-commercial sources, the supply chain, customers, and governments.

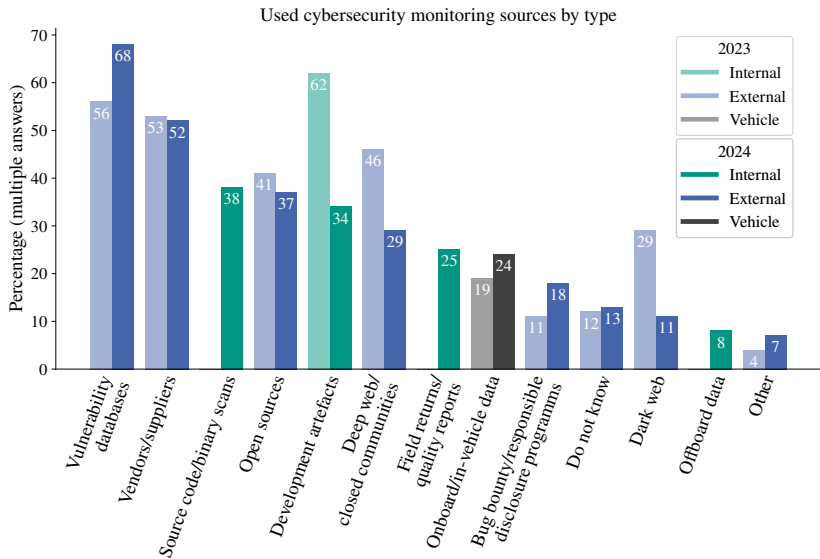


Figure 2.15: Development of sources used for cybersecurity monitoring 2023 and 2024 (adapted from [Her+24]). Differentiation in internal, external and in-vehicle sources was added.

In 2023, development artifacts were the most important source worldwide in the industry (cf. Figure 2.15 on page 51), while closed communities such as ISAC⁶⁵ (cf. Section *External sources* on page 80) were the most important source in the USA and Japan [Min23a]. However, organizations that rated themselves as very mature used hacker forums, researchers, vehicle data (twice as often) and bug bounty programs (three times more often, cf. Remark 4.2 on page 84) disproportionately often [Min23b]. In 2024, external sources, especially vulnerability databases and suppliers, became the primary reference [Her+24], and usage of in-vehicle data is growing slowly. However, the survey also highlights the continued relevance of internal sources, considering the distinction between development artifacts, source code scanning, and the inclusion of field quality reports.

⁶⁵ Information Sharing and Analysis Centers

Evaluation, analysis and incident response

The generated events in cybersecurity monitoring shall be analyzed to identify weaknesses (event evaluation, clause 8.4). Vulnerability analysis (8.5) decides whether the weakness is a vulnerability, in which case a risk assessment (cf. Section 2.3.1 on page 40) is performed. Note, that the weakness is an optional input to vulnerability analysis, indicating that vulnerabilities may be found independent of previous identification of weaknesses. Finally, in vulnerability management (clause 8.6), countermeasures are selected according to the risk of the vulnerability, or residual risks are accepted. The remedial actions are documented and implemented in an incident response plan in the incident response step (clause 13.3, cf. Remark 2.5), which is part of the operations and maintenance.

Remark 2.5: Incident response plan (ISO/SAE 21434)

The incident response plan defines for each incident [ISO21]:

- a) remedial actions, determined by vulnerability management
- b) communication plan to relevant internal and external parties, including development of specific information for the audience
- c) assigned responsibilities for remedial actions
- d) procedure for recording new relevant cybersecurity information
- e) method for determining remediation progress
- f) closure criteria
- g) closure actions

Incident handling is to be invoked as part of vulnerability management, with the vulnerability analysis being an optional input (cf. Figure 2.14 on page 49). The response plan shall be created for each incident and contains further documentation, such as responsibilities, communication strategies, and criteria for determining the progress and closure of the case (cf. Remark 2.5 on page 52). Finally, clause 13.4 mandates that the development of updates must conform to ISO/SAE 21434 as well.

3 Ensuring vehicular cybersecurity during operations

3.1 Idea of this thesis

This dissertation focuses on maintaining the cybersecurity of the vehicles of a particular OEM during their operation, targeting the tasks resulting from UN R155. The assumed vehicles feature modern E/E architectures, including SOA-based software architectures, high performance hardware with generic execution capability, communication with backend services and automated driving functions.

Besides technological aspects, this work addresses the required processes (cf. Research Question 1) with two ideas: a process for cybersecurity operations (cf. Definition 1.6 on page 8, ① in Figure 3.1 on page 57) and support for the development of in-vehicle monitoring (cf. ② in Figure 3.1 on page 57). For both process-related ideas ISO/SAE 21434 is the starting point (cf. Section 2.3 on page 39), because it is currently the most important process to meet UN R155's requirements.

A comprehensive data perspective is the primary technological approach of this work (cf. ③-⑤ in Figure 3.1 on page 57), which addresses cybersecurity monitoring (cf. Task T1 and Research Question 2). A usable data basis is the prerequisite for successful analysis and identification of attacks, threats and vulnerabilities (cf. Task T2), which in turn constitutes the basis for determining suitable responses. The second idea addresses the technical realization of responses in vehicles (cf. Research Question 3, Task T5 and ⑥ in Figure 3.1 on page 57)

3.1.1 Processes

① **Process(es) for cybersecurity operations activities**

Once the vehicle is produced, the V-model ends. What happens next? Processes that maintain the cybersecurity of the vehicle fleet are necessary. While the development of updates is a potentially reduced but conventional (as for the development of an entire vehicle) software development process (cf. Section 2.2.4 on page 32), this work's idea is to define a new process (or several) focusing entirely on cybersecurity operations (cf. ① in Figure 3.1 on page 57). The activities of the processes should be aligned with development activities, including development of updates. Such new processes should consider at least all activities defined as tasks in UN R155, including monitoring (cf. Task T1), analysis (cf. Task T2), updating risk assessments and cybersecurity measures (cf. Task T3 and Task T4), and timely responses (cf. Task T5) to incidents and vulnerabilities. The automotive-specific challenges due to distributed development should be taken into account when defining the activities and their interfaces.

② **Procedure to specify monitoring requirements**

To procedurally support in-vehicle monitoring, the specification of the according requirements should be anchored in the development process (cf. ② in Figure 3.1 on page 57). The cybersecurity monitoring requirements should ensure adequate observability of the risks identified in the TARA. This is based on the assumption that a great deal of data could be collected - but what is relevant is determined by the existing cybersecurity risks of a specific architecture. Mistakes in requirements definition are known to be very costly, as they entail a lot of rework when they are rectified later. Therefore, the requirements definition phase should be methodically supported. Since defining monitoring requirements in the development process is a procedure to support cybersecurity operations, the idea addresses both Research Question 1 and Research Question 2.

3.1.2 Technologies

③ In-vehicle monitoring of cyber and physical aspects

For a comprehensive data perspective, the first aspect is to collect relevant data within the vehicle (cf. ③ in Figure 3.1 on page 57). Accordingly, an in-vehicle monitoring system should be in place that can observe cyberattacks on future software-defined, highly automated vehicles.

As the vehicle is a cyber-physical system, in-vehicle monitoring should account for both the cyber and physical aspects of a vehicle's cybersecurity. The cyber side refers to the monitoring of the E/E architecture, essentially networks, hosts and software. However, vehicles interact with their environment with sensors and actuators. Thus, cyberattacks could have physical causes or constraints (sensor or environmental manipulation, time, weather) and effects. A fleet-wide attack based on the same vulnerabilities could result in similar physical effects, and possibly temporal correlations. Thus, monitoring should take the physical dimension into account.

④ Fleet-wide collection

Each vehicle in the OEM's entire fleet should transmit its collected data to a corporate backend system via the existing OTA connectivity (cf. ④ in Figure 3.1 on page 57). Cybersecurity of the data in transmission should be ensured in particular. Only authenticated vehicles should be able to send their data to the backend. A suitable data format is required to transmit cybersecurity information, including traceability to the data sources.

⑤ Connection with additional information

Once collected, the next step is to link the in-vehicle data from the fleet with additional information in the backend. Today, the industry uses manufacturer-internal

(e.g. development artefacts) or external (e.g. public databases) sources more often than vehicle-internal data for cybersecurity monitoring (cf. Figure 2.15 on page 51). The monitoring of incidents ("occurrence in a system", cf. Definition 1.2 on page 6) requires in-vehicle data. Accordingly, cybersecurity monitoring should create a holistic data basis that integrates all three types of sources: vehicle-internal, manufacturer-internal and external data (cf. ⑤ in Figure 3.1 on page 57).

Cybersecurity monitoring includes the pre-processing and formatting of cybersecurity information for further usage, such as the identification of attacks (cf. Definition 1.7 on page 8). To make the collected data usable, pre-processing and formatting should involve building a semantic connection between all collected data, and long-term storage. Though this could build on databases with standardized query languages, cybersecurity deals with the unforeseen (cf. safety versus security, Section 1.4 on page 11), which is why flexibility should be also accounted for.

⑥ Immediate response over the air

The second technological idea (cf. ⑥ in Figure 3.1 on page 57) is the vehicle's ability to respond (cf. Research Question 3 on page 13 and Task T5). The focus is on the *timeframe*: reactions should be possible immediately, if this is necessary to ensure the passengers' physical safety.

However, because updates need to be homologated (cf. Section 2.2.4 on page 32), updates alone are insufficient to meet the required responsiveness. Therefore, besides OTA updates, vehicles could be informed of necessary reactions using a secure backend connection. The vehicles should then execute these automatically to adapt to attacks or new risks. Similar to the MRM for the safety of automated driving functions, the aim is to reduce the cybersecurity risks by reactions (cf. Definition 1.8 on page 8). A state of minimal cybersecurity risk would be the result of a successful, appropriate response.

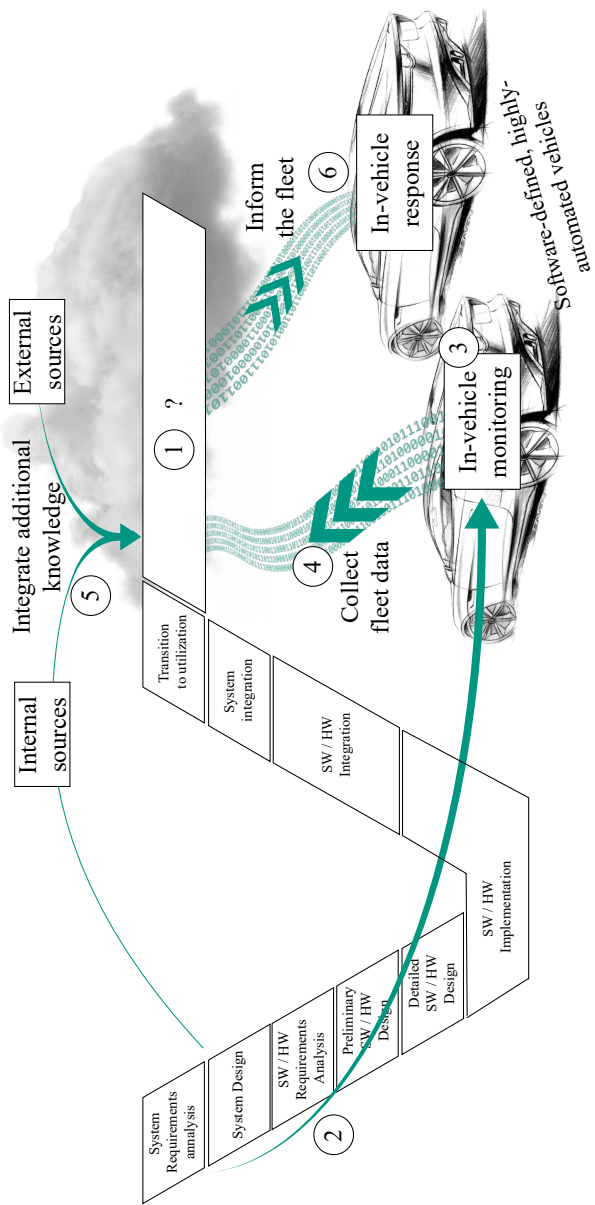


Figure 3.1: Overview of this dissertation's idea.

3.2 Requirements

The outlined idea is supported by the following requirements (Rq.), organized according to the six main approaches of the work.

Process(es) for cybersecurity operations activities:

- Rq . ①-1 The operations process should support all tasks defined in UN R155.
- Rq . ①-2 Automotive-specific constraints due to distributed development should be taken into account.
- Rq . ①-3 The operations process should be aligned with the development process, in particular for updates.

Procedure to specify monitoring requirements:

- Rq . ②-1 The development process should contain the necessary phases to specify requirements for in-vehicle monitoring.
- Rq . ②-2 A procedure to specify monitoring requirements should focus on observing the prevalent cybersecurity risks.

Monitoring of cyber and physical in-vehicle aspects:

- Rq . ③-1 In-vehicle monitoring should cover the cyber aspects of vehicles.
- Rq . ③-2 In-vehicle monitoring should cover the physical situation.
- Rq . ③-3 The characteristics of software-defined vehicles should be taken into account.

Fleet-wide collection:

- Rq . ④-1 The currently relevant data within each vehicle should be collected in a backend.

- Rq . ④-2 The transmission to a backend must be secure to ensure no information is lost or can be manipulated.

Connection with additional information:

- Rq . ⑤-1 Relevant information from the vehicle manufacturer shall be associated with the vehicle's internal data.
- Rq . ⑤-2 Relevant public or manufacturer-external data should be linked with the vehicle's internal data and manufacturer-internal data.
- Rq . ⑤-3 The linked data should be stored for long-term usage.
- Rq . ⑤-4 Data storage should include semantics to enable searching and analysis.
- Rq . ⑤-5 The semantics of the data association should be flexible enough to allow to incorporate new information.

Immediate response over the air:

- Rq . ⑥-1 There should be a secure transmission channel from the backend to the vehicles to deliver cybersecurity responses.
- Rq . ⑥-2 The vehicle shall be able to carry out adequate cybersecurity reactions automatically without introducing significant safety risks.

4 Current state of cybersecurity operations for vehicles

This dissertation explores various aspects of cybersecurity operations. The central ideas of this thesis (① to ⑥) address the three research questions resulting from the manufacturer’s tasks according to UN R155 (cf. Figure 4.1). The previous chapter set out requirements for these ideas (cf. Section 3.2). Subsequently, relevant literature and material from the industry were identified, which capture the state of the art in science and technology. In summary, the ideas and requirements involve procedural and technological aspects, which the structure of this chapter (cf. Figure 4.2) reflects.

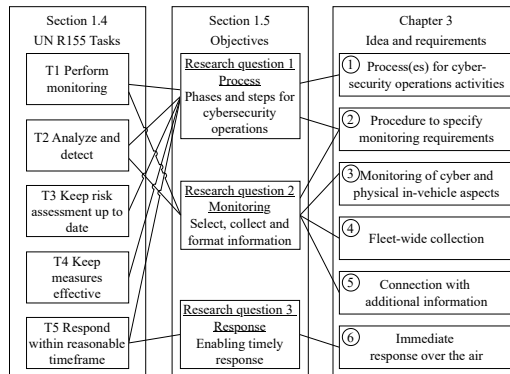


Figure 4.1: Mapping the UN R155 tasks and research questions of this thesis to ideas and requirements.

As idea ① shall unite UN R155’s tasks procedurally, standards from IT cybersecurity and automotive-specific research that applies ISO/SAE 21434 or focuses

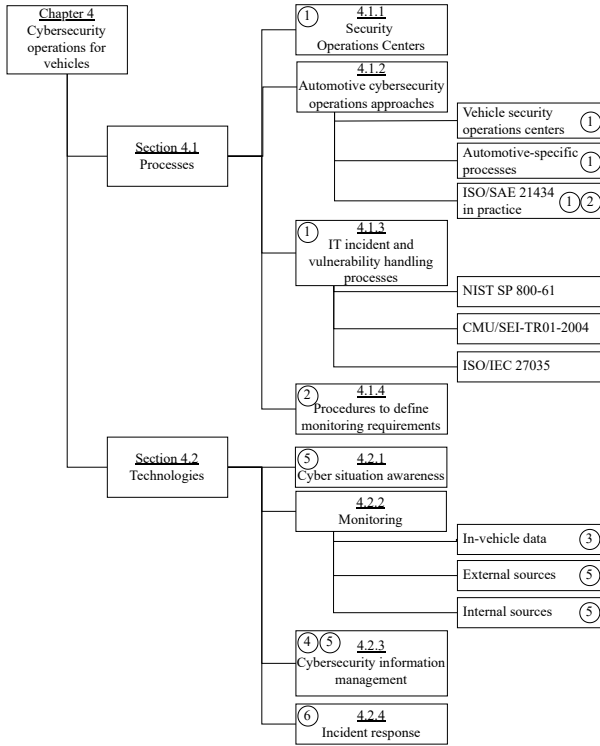


Figure 4.2: Structure of the considered related work in cybersecurity operations and alignment with this thesis' ideas.

on cybersecurity operations are relevant. Also, for the requirements regarding incident response, the generic approaches from IT processes are contrasted with the vehicle-specific solution approaches (cf. ⑥). Approaches related to idea ② cover procedural aspects (being development methods) but are closely linked to in-vehicle monitoring technology (cf. ③ and Section *In-vehicle data* on page 76). As the E/E architecture of SDVs becomes more similar to generic IT networks, methods from IT network security monitoring are of significance. The industrial statistics and ISO/SAE 21434's recommendations reveal that external and internal sources are highly relevant for monitoring. Accordingly, concerning ⑤, publications that involve OEM internal and external information for monitoring are of

interest. In addition, works from CSA (Cybersecurity Situation Awareness) and information management were examined, as these discuss required information for comprehensive understanding, collection, and the combination of information (cf. ④ and ⑤). The importance of CSA for this thesis stands out, as holistic monitoring seeks to perceive the cybersecurity situation as completely as possible. The experience gained in this field provides a further argument for combining different sources.

4.1 Processes

4.1.1 Security Operations Centers

Considering vehicle cybersecurity operations, the literature discusses majorly two concepts: *SOC*¹ and *SIEM*². While SIEM systems are technology (tools) (cf. Definition 4.8), SOC also include processes and personnel.

Definition 4.1: SOC

"The Security Operations Center (SOC) represents an organizational aspect of an enterprise's security strategy. It combines processes, technologies, and people to manage and enhance an organization's overall security posture. This goal can usually not be accomplished by a single entity or system but rather by a complex structure. It creates situational awareness, mitigates the exposed risks, and helps to fulfill regulatory requirements. Additionally, a SOC provides governance and compliance as a framework in which people operate and to which processes and technologies are tailored." [Vie+20]

¹ Security Operations Center

² Security Information and Event Management

Strong revenue growth for vehicle SOC³s, i.e. VSOC³, is expected to reach around one billion US\$ per year by 2030. For cybersecurity processes and solutions in general, the estimates amount to an annual growth rate of 4% between 2020 and 2030 [Bur+20]. However, the introduction of VSOCs and cybersecurity operations pose automotive-specific challenges. These include the scale of assets (millions of vehicles, long lifetime, worldwide, 24/7), restricted physical access and remote connectivity, potentially modified hardware and software due to the right to repair, and limited storage capacity within the vehicles [HBH23].

Capabilities

Since SOC³s combine people, processes, and technology, the implementations vary in corporate contexts, as suitable personnel structures depend on the size and structure of the company [KPZ23]. For VSOCs, complete or partial outsourcing to third-party providers, integration into the manufacturer's existing IT SOC, or the establishment of a dedicated VSOC within vehicle development or quality management are discussed [Bur+20]. Due to different data protection requirements and software features, operating several VSOCs is an option to achieve global coverage [Bur+20].

The processes in SOC³s are usually derived from incident handling processes [Vie+20] (cf. Section 4.1.3). Common capabilities (provided services) of SOC³s are security monitoring, analysis, incident management, threat intelligence (cf. Section *External sources* on page 80), vulnerability management, threat hunting, and cybersecurity information management [Van16]. Besides, situational awareness, communications and training, and security architecture administration (maintenance of the detection and protection technologies) can be capabilities [KPZ23]. However, the compulsory capabilities of SOC³s are real-time observation of incoming information, triage, analysis, and incident response [KPZ23].

³ Vehicle Security Operations Centers

Roles

ISO/SAE 21434 and UN R155 require defined roles and responsibilities and the trained personnel (cf. [RQ-05-03] and [RQ-05-07] in ISO/SAE 21434 and as part of the CSMS, cf. Definition 1.4). Consequently, these requirements also apply to vehicle cybersecurity operations. For detection and analysis of incidents, usually a multi-level escalation structure (e.g. three levels) is followed: *Tier 1* analysts consult a *Tier 2* analyst when a high criticality of an incident is detected, who in turn can escalate the analysis case to a *Tier 3* analyst [Vie+20; Olt18]. The analysts are led by a SOC manager and incident response coordinator [Olt18] (cf. Figure 4.3).

Tier 1 is a *Triage specialist*. This role is responsible for receiving cybersecurity information, confirming relevance (or withdrawing false positives) and assigning a criticality. For this purpose, alarms from detection systems and raw data are reviewed. Detection systems are, for example, SIEMs (cf. Figure 4.3).

Tier 2 are *Incident responders* that analyze the triaged incidents in detail to design and implement a response strategy. This involves identifying affected systems to understand the attack's scope and correlating the data with threat information (cf. Section *External sources* on page 80) to understand attacker targets and behavior. If the Tier 2 analysts cannot solve the case (fast enough), it is escalated to the most experienced analysts.

Tier 3 are the *threat hunters*. In addition to handling the most critical incidents, they are responsible for carrying out vulnerability assessments and penetration tests to anticipate vulnerabilities and identify new threats. Their expertise helps to improve the security monitoring.

Other technical roles in a SOC are *malware analysts*, who reverse-engineer malware, and *threat intelligence analysts*, who analyze external threat intelligence

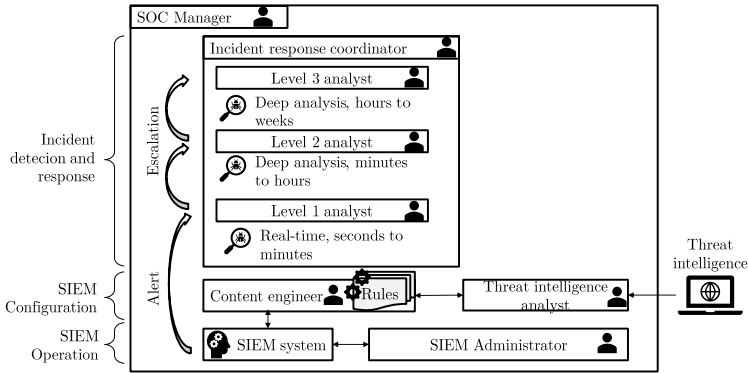


Figure 4.3: T-Systems VSOC approach (derived from [Olt18]).

sharing sources to create and prepare indicators that can be correlated in incident response. [Vie+20]

4.1.2 Automotive cybersecurity operations approaches

Vehicle security operations centers

In addition to commercial providers for VSOCs such as Upstream [Ups23]⁴, and T-Systems [T-S24], different perspectives are discussed in the literature. More overarching, Oka [Ken21] proposes three aspects for cybersecurity operations: Release management, continuous monitoring, and secure updates. Continuous monitoring revolves around introducing a VSOC and accompanying process, which is provided with threat and vulnerability information from various sources. These include the Auto-ISAC⁵, bug bounty programs (cf. Remark 4.2), a SBOM⁶,

⁴ Renault, Volvo, Hyundai, BMW and Ford Trucks invested or partnered with Upstream [Beu19; Ven21]

⁵ Automotive Information Sharing and Analysis Center

⁶ Software Bill of Materials

and on-field reports. Release management involves generating a SBOM by scanning the code and assigning known vulnerabilities to the open-source software packages found (cf. Remark 4.3). Recently, a publication specified several components and their constraints to implement a VSOC that includes information from vehicles, threat information and the connected vehicle ecosystem [May+24].

Barletta et al. [Bar+23b; Bar+23a] present a VSOC approach, and a data model based on a given attack database (cf. [SDK19]). The VSOC [Bar+23b] focuses on the implementation of custom rules within a SIEM to detect Denial-of-Service and fuzzing attacks, that were simulated on in-vehicle CAN communication. The data model aims to create an attack graph using the multi-step attacks of the given database.

A larger group of researchers describes in several publications [LSS19; LSS20; Mey+20; H  c+23] an approach for a VSOC, which they call *Automotive Cyber Defense Center* (ACDC). The authors propose a multi-level concept consisting of a fleet-level SOC connected to an in-vehicle (edge) defense center. Monitoring and analysis takes place in the edge and the ACDC, to enable both autonomous and cloud-triggered reactions (cf. Section *Responses in vehicles* on page 92). The ACDC shall also incorporate external information from researchers, suppliers or threat exchange platforms for the investigation of incidents.

Automotive-specific processes

Automotive-specific publications propose few specific processes, but usually refer to established ones from IT [HG18; LSS20; Pia22]. In particular, NIST SP 800-61 [NIS12] (cf. Section *NIST SP 800-61* on page 70) is proposed as the procedural basis for a VSOC (cf. Section 4.1.2) [LSS20] and for synchronization with safety monitoring (ISO/PAS 21448 [ISO22] SOTIF) [Pia22]. Besides safety, automotive-specific process-related aspects are the integration of the dealer and OEM help desk in the reception of information on possible vulnerabilities and attacks (cf. Figure 4.4), and the interaction with suppliers [HG18].

Remark 4.1: Support organization

The OEM infrastructure for receiving problem reports and complaints, including the dealer network, help desk and other interfaces to the public, such as public relations, is referred to as the *support organization*.

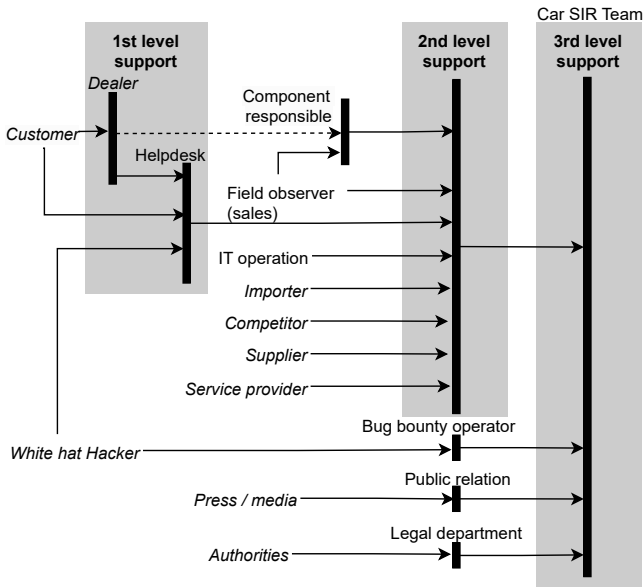


Figure 4.4: Support organization and information flow for registering car security incidents at an OEM (derived from [HG18]). External reporters are written in *italics*.

Furthermore, process models for the entire lifecycle based on DevOps are considered [Sch+20b; Dob+23]. According to the authors, fundamental to DevOps is an SDV architecture that can be updated and reconfigured [Sch+20b], and a model-based TARA [Dob+23]. During operations, the multi-level threat modeling created in the design phase could be used for risk assessment and to search indicators for malicious activity. An industrial publication by Motional [Wil23] outlines a complete development process, with the operations phase based on

NIST SP 800-218 [NIS22]. Here, the focus is on the application of TARA for prioritizing vulnerabilities and defining document templates to achieve compliance with ISO/SAE 21434 and UN R155. Monitoring and incident response plan are defined in a preceding foundation phase.

ISO/SAE 21434 in practice

A number of works analyze ISO/SAE 21434 and highlight practical implications. In particular, the application of TARA [EJ22] and respective methods are discussed. Challenges that are addressed are, for instance, the application of TARA to an entire vehicle, cooperation with suppliers, and sharing of best practices. One suggestion is to perform TARA at different levels of abstraction [Lie21; Lie+24; Dob+23; Gri+23b]. Breaking down and refining the TARA into different *TARA levels* reflects the joint development activities of the OEM and suppliers. While the OEM carries out the TARA for a specific feature at vehicle-level (feature in the context of the connected vehicle with external interfaces), the suppliers are responsible for analyzing their components (e.g., specific ECUs) in detail. The different TARAs are denoted, for example, by level 0 (vehicle-level), level 1 (component-level), etc., whereby vehicle-level corresponds to the understanding of the concept phase (Clause 9, cf. Figure 2.10), and the other levels 1- n correspond to the iterative refinements in the product development phase (Clause 10, cf. Figure 2.10).

Furthermore, sharing TARA artifacts with suppliers is discussed as part of the cybersecurity interface agreement to analyze attack paths [KA21]. Others suggest to share the damage scenarios with suppliers to improve the impact rating [Gre+22]. Also, improvements to the risk rating scheme in ISO/SAE 21434 are discussed, such as using attacker types to classify capabilities, and changed attack potential ratings [Gre+22]. Another improved attack feasibility rating strategy considers rating per attack step in the attack paths, including the influence of security measures [Pla+21]. Other approaches focus on the automation of threat modeling using attacker capabilities and attack trees or graphs [Zel+22; Chl+23; DSK21].

Few works address the cybersecurity operations activities. The interaction with suppliers when exchanging incident information could be scoped with the cybersecurity interface agreement [KA21]. Recently, a publication outlined that the use of CVSS for attack feasibility rating and CWE⁷ for attack paths could support vulnerability management [Ill+24]. By mapping components to CVEs, new vulnerabilities can be prioritized using the vulnerabilities' CVSS and violated cybersecurity properties of the affected component. Potentially missing steps in the component's attack path can be identified.

4.1.3 IT security standards and guidelines

As cybersecurity has played a role in the IT for much longer than in the automotive sector, there is a comprehensive body of standards and process models, the summary of which would go beyond the scope of this thesis. The focus is therefore on standards that deal with monitoring and response capabilities. In this scope, process models for incident handling have been present over the last 20 years, including NIST⁸ SP 800-61 [NIS12], CMU/SEI-2004-TR01 [Alb+04], and ISO/IEC 27035 [ISO16b]. Besides these specific procedures, military thought patterns like the *OODA*⁹-loop are common for SOC's [Hus+22; Osi07]. OODA originally describes decision and action loops in air combat, but is used in incident handling as a model for fast, data-based decisions.

NIST SP 800-61

The NIST SP 800-61 Computer Security Incident Handling Guide [NIS12] describes a four-step process (cf. Figure 4.5a). *Preparation* includes preventive

⁷ Common Weakness Enumeration (cf. glossary, page 325)

⁸ National Institute of Standards and Technology

⁹ observe, orient, decide, act

measures to avoid incidents through application of risk assessment, implementation of security measures, user training, and preparations to handle incidents. Resources and tools are prepared, including contact information of relevant people, forensic software, and technical resources for analysis. The technical information should include port lists, documentation for OSs, applications, protocols, and cybersecurity controls, network diagrams and lists of critical assets, baselines of the expected system behavior or cryptographic hashes of critical files.

Detection & analysis considers sources for potential incidents, e.g. IDS alerts, logs, and public vulnerability information. The analysis includes validation and initial analysis, prioritization, documentation, and notification of relevant actors. The initial analysis should determine a rough scope of the incident to appropriately prioritize and, if necessary, analyze the incident further. The third step includes *containment* to reduce the damage, *eradication* to eliminate the incident, and finally *recovery* to return to normal operations. A re-iteration of the analysis phase is often required to resolve all problems. *Post-incident activity* involves lessons learned meetings and collecting subjective and objective measures (e.g., number of incidents handled, time per incident). The identified improvements for incident handling are fed back into *preparation*.

CMU/SEI-2004-TR01

In CMU/SEI-2004-TR01 [Alb+04] (cf. Figure 4.5b), the post-incident analysis takes place in the *prepare, sustain and improve* step. The subsequent *protect* step is similar to the *prevent incidents* sub-step in the *preparation* phase of NIST SP 800-61, i.e., ensuring that industry best practices are followed. The *detect events* phase monitors infrastructure and collects external reports, and forwards potential incidents. An initial analysis consisting of validation, filtering and prioritization takes place in a separate *triage events* step. The *respond* phase is iterative with three sub-phases, until the incident is closed. Main goals are to understand the incident, contain it and recover to normal operating conditions, similar to the third phase in NIST SP 800-61. The iterative response phase consists of a detailed analysis, where additional information may be collected, and the response will

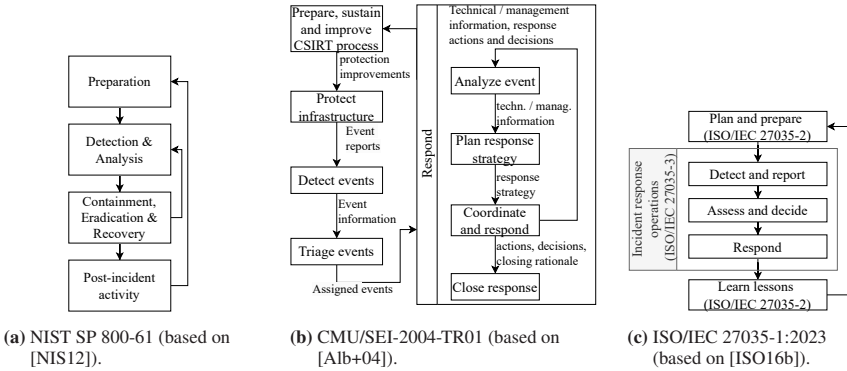


Figure 4.5: IT security incident management and handling processes.

be planned, coordinated and executed. Relevant information for post-mortem analysis is fed back to the *prepare, sustain, and improve* phase.

ISO/IEC 27035

ISO/IEC 27035 "Information security incident management" consists of three parts. Part 1 outlines the principles and process, having a total of five phases (cf. Figure 4.5c). Part 2 addresses the phases of *plan and prepare* and continuous improvement (*learn lessons*). Part 3 details the three phases for incident response operations, including monitoring and threat intelligence gathering (*detect and report*), to analysis and management (*assess and decide*) and response execution (*respond*). In particular, numerous response activities (cf. Section *General approaches* on page 94) and monitoring sources are considered. With respect to monitoring, technical (e.g., IDS, anti-virus, log analysis and SIEM tools), people (e.g., users, customers), and organizational measures (e.g., IT help desk, service providers, threat feeds) are mentioned. The standard is comprehensive in its descriptions, but does not provide a concrete, assessable list of requirements.

4.1.4 Procedures to specify monitoring requirements

Monitoring networks involves a variety *sensors*, that are placed on selected to generate CSI¹⁰ from different perspectives. Similar to sensors for the driving functions (camera, GNSS¹¹), these sensors perceive different aspects of their (network) environment. The best-known type of sensors are IDS. However, further technical measures are applicable (cf. Section *In-vehicle data* on page 76).

Definition 4.2: Cybersecurity monitoring sensor

A cybersecurity monitoring sensor is a component, that generates cybersecurity information from observing the operation of a system.

From a procedural perspective, the selection and placement of sensors in the network is critical. The different sensor types and their placement (allocation as software on existing hardware or integration of specific hardware) in the network determine their perception. To stay with the analogy of sensors for driving functions, sensors must be properly combined to provide the desired function (cybersecurity) within the given boundary conditions (architecture, interfaces).

A method to specify monitoring requirements needs to address two questions: *what* should be observed, and *how* this can be technically implemented. A basic recommendation is to consider each CMS¹²'s vantage, not to cause either strong redundancy or invisibility of attacks [Col14].

A generic method for placing sensors is the *ACF*¹³ [SS14], a four-step process to arrive at suitable measures (cf. Figure 4.6). Here, *what* is to be observed are the threats identified as particularly critical by calculating a risk value. For this purpose, organization-specific threats are identified, focusing on worst-case scenarios and critical assets. The second step is to prioritize the identified threats based on calculated risk, which is determined by impact and probability of each

¹⁰ cybersecurity information

¹¹ Global Navigation Satellite System (cf. glossary, page 325)

¹² cybersecurity monitoring sensor

¹³ Applied Collection Framework

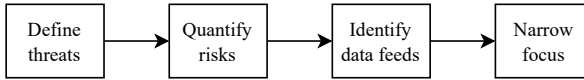


Figure 4.6: The Applied Collection Framework (derived from [SS14]).

threat. Steps three and four address the *how*. Analyzing how the highest-risk threats would manifest on the network identifies potential data sources (CMS), which can include data of all monitoring domains (cf. Section *In-vehicle data* on page 76). Step four evaluates each source for its value and resource consumption. A cost-benefit analysis considers hardware, software, staff needs, and documented usage of the collected data in previous investigations. Specific data elements are then chosen within each source, and a data retention policy is defined. Collection of network traffic, for example, might involve specific ports, protocols, or traffic types. The result is a granular list of data elements to be collected by CMS, guiding network and security teams for data collection configuration.

More elaborate approaches use statistical methods to optimize CMS positions within a network. For example, recorded data of CMS can be used to measure the frequency of indicators in normal data, and generated attack sequences on a network model yields the observability of the current placement [HWH19]. In this case, the goal of monitoring (what) is to detect selected techniques from MITRE¹⁴ ATTACK¹⁵ (cf. Section *External sources* on page 80). Another approach [MBL08] uses a Bayesian network for optimizing intrusion detector placement by assuming attacker probabilities. In IT, using MITRE ATT&CK® as a coverage metric has become established. Products such as SIEM systems and HIDS¹⁶ advertise their ability to detect a certain percentage of techniques, and decision-makers assume that 100% coverage equals 100 % secure. However, a recently published study [Vir+24] highlights that this is a misconception, especially since detection does not take place at technique level, but can only detect some of its manifestations - but not necessarily all of them.

¹⁴ cf. glossary, page 327

¹⁵ Adversarial Tactics, Techniques, and Common Knowledge

¹⁶ Host-based Intrusion Detection System (cf. glossary, page 326)

4.2 Technologies

4.2.1 Cyber situation awareness

The tasks of SOC's (cf. Section *Capabilities* on page 64) involve several decisions, such as escalation in triage and selecting suitable responses. The decisions require a basis for decision-making: the security situation must be understood, and the decision makers must "know what is going on" [Fra+22]. This understanding is referred to as CSA. For some authors [Vie+20; KPZ23], the capability of creating situational awareness is considered a defining aspect of SOC's (cf. Definition 4.1).

CSA conceptually divides understanding the cybersecurity situation into three steps *perception of the environment*, *comprehension*, and *projection in the near future* [End88; Kom+18]. Perception should incorporate different data sources, covering both *technical* awareness (network situation) and *strategic* awareness (the threats to and the mission of an organization) [Fra+22; Ali16]. Understanding involves interpreting and combining data into knowledge. Due to the necessary communication within the team, for example, this also entails socio-cognitive challenges [Fra+22]. However, this thesis focuses on the technological aspects.

In this thesis, two technologies contribute to CSA: monitoring and information management. The goal of monitoring is to provide the necessary information for perception. Information management facilitates comprehension of the situation by preparing and presenting the collected data so that users, particularly the VSOC staff, can act on it. In the automotive domain, vehicle-internal sources can create technical awareness. External sources such as CTI¹⁷ and the knowledge from the manufacturer cover the strategic perspective.

¹⁷ Cyber Threat Intelligence

4.2.2 Monitoring

The following section discusses sources for cybersecurity monitoring. For this purpose, the sources are structured according to in-vehicle, OEM-external and OEM-internal sources (cf. Figure 4.7), corresponding to those used in the industry (cf. Figure 2.15). In addition to the IT-proven monitoring technologies for the cyber aspects, the internal vehicle data sources also include work considering monitoring the physical aspects.

In-vehicle data

a) Network security monitoring From a technological perspective, the cyber aspects of in-vehicle data correspond to the IT cybersecurity field of NSM¹⁸. While the use of in-vehicle data is not yet well established in the industry, research concentrates on this field, particularly on IDS. In a growing market for automotive cybersecurity, several vendors are offering IDS products [Per24], which indicates a belief in the need for such solutions. However, besides IDS, several other measures can provide a technical understanding of the network situation. Generally, CMS are differentiated by their *monitoring domain* (network, host, and service), *vantage point*, and *action* [Col14; GPS20] (cf. Table 4.1). The monitoring domain defines the observed aspect of the system. CMS's actions involve collecting (C), alerting (A), or blocking (B) data. The position in the considered network determines the vantage of a CMS.

CMS of the network domain usually monitor OSI layers 1-4 from the physical layer to the TCP or UDP connection. Sensors on the service domain consider data of specific applications on layers 5-7. OS typically implement the OSI layers 2-4 (cf. Figure 2.5) and provide an interface to the respective protocols. However, as the OS is not inherently a layer of the OSI model, host-domain sensors monitor further aspects besides communication, including access to resources, writing files, or starting processes. Another approach only differentiates network

¹⁸ network security monitoring

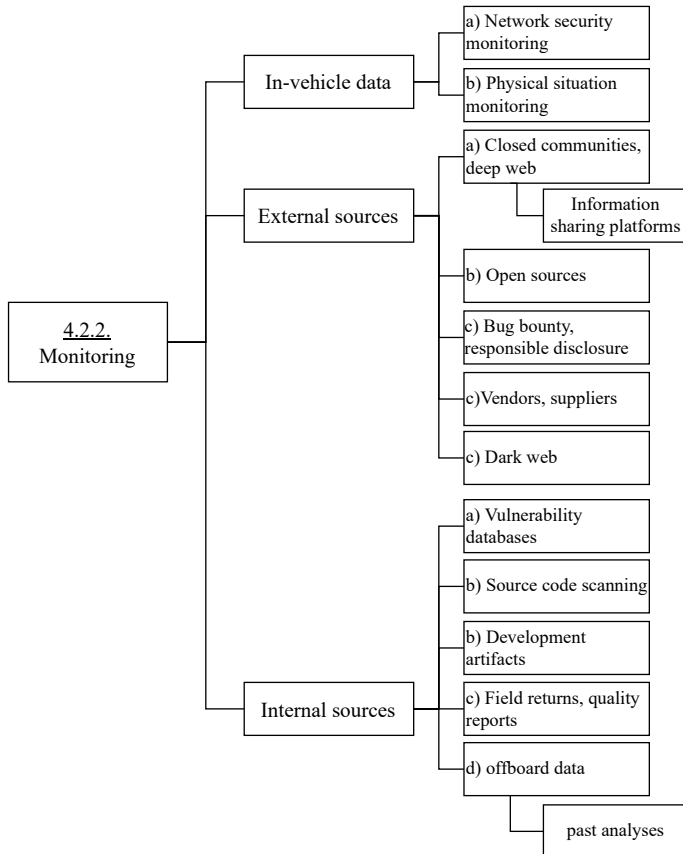


Figure 4.7: Overview of sources for monitoring, derived from Figure 2.15.

monitoring and endpoint protection [Arm+24] (combining hosts and services into one domain) but additionally recommends physics-based monitoring. In general, any security measure (cf. Section C.3) can be a CMS if it traces its decisions. Other sources, such as OS and applications, are CMS if their logs and interactions are collected accordingly (cf. Section 4.2.3).

Table 4.1: The monitoring domains of CMS, their action (superscript C: collect, A: alert, B: block), and approximate categorization in the OSI model.

Domain	Examples	OSI layer
Service	Application interaction logs ^C and status ^C	5-7
Host	HIDS ^A , OS logs ^C , host state ^C , access control ^B	(2-4)
Network	NIDS ^A , IPS ^B , firewall ^B , network flows ^C , packet capture ^C	1-4

As CAN and other bus systems lack inherent security mechanisms, they are vulnerable to several attacks, including denial-of-service (sending only zeros or provoking bus-off error state), spoofing (sending arbitrary messages with identifiers of other participants), or tampering (compromising the original contents of a message). Consequently, a vast amount of research in NSM discusses network-domain CMS, with a focus on IDS (action: alert) for CAN [LM23; Rum+20; Wu+20; Web19]. A significant fraction of the research on IDS for CAN are signal-based IDS, detecting deviations of individual signals from normal behavior, which is a difference from usual Ethernet frame-based IT IDS. Besides IDS and IPS (action: block), collection CMS for CAN are considered [Gör+22; Wu+22; Gri+20]. For SOA and Ethernet, several works address attack detection, firewalls, and access control mechanisms [De +24; GWS18]. Generally, abnormal messages on the vehicle’s internal networks, status information and other data from the ECUs, and alarms from IDS within the vehicle could be collected [Olt18].

From industrial perspective, AUTOSAR standardizes the processing of AUTOSAR SEV¹⁹ (cf. Section *Protocols and standards* on page 89). As a result, several AUTOSAR components, including the Communication Management, Diagnostic Management, and Firewall, report SEVs [AUT23b], which provides different vantage points to monitor AUTOSAR-based systems. Using access control specified through the IAM²⁰ component [AUT18], and logging with DLT²¹ could provide further insights into the host and service domains. A publication combined Linux access control with the SEVs from key management and cryptographic services and modified process execution in an IDPS²², providing alerting and blocking of unauthorized processes in AUTOSAR systems [Val+24].

b) Physical situation monitoring First and foremost, observing the physical situation, i.e., the vehicle’s environment, is a core issue of automated driving. Especially for higher automation (SAE Level 3 and higher [SAE18]), in which drivers may perform other activities while driving, precise knowledge of the current conditions (COD) compared to the permitted operating conditions (ODD) is essential. Accordingly, several works address taxonomies and domain-specific languages to specify operating conditions and different approaches to monitor them [Cha+23; SB24; ISO23a].

However, also physical aspects are considered for cybersecurity, particularly within IDS. For example, sensor data for automated driving, such as a camera or rain sensor, are fed together with CAN frames into a machine learning-based IDS to detect abnormal frames [Jia+20]. Others use throttle and brake pedal positions from a simulation environment to train an IDS [Mic+24]. Besides low-level physical signals, a high-level vehicle state, expressed in a state machine, was used for relevance filtering of identified anomalies by different sensors [Was+17]. In addition to the IDS-focused work, physical aspects are also partly considered within incident handling. One example is the combination of (cybersecurity)

¹⁹ Security Event

²⁰ Identity and Access Management (cf. glossary, page 326)

²¹ Diagnostic Log and Trace (cf. glossary, page 325)

²² Intrusion Detection and Prevention System (cf. glossary, page 326)

incident handling with the handling of SOTIF incidents [Piq22], as understanding the environment is required for the analysis of functional misbehavior. In addition, the use of safety and physical-related information is also discussed in the literature for VSOCs [May+24; LSS19; GSS21].

External sources

External sources like governments, researchers, press news, blogs or commercial sources considered for monitoring usually provide *threat information*:

Definition 4.3: Threat information

"Analytical insights into trends, technologies, or tactics of an adversarial nature affecting information systems security." [NIS19]

Threat information shall help to defend against certain attackers or groups. A common term to describe this type of monitoring is *CTI* (*Cyber Threat Intelligence*). This term implies that trends, technologies and tactics are not "raw", e.g. packets from a network, but processed ("intelligence"). Outcomes of CTI include plain text reports, threat models formalized as TTP²³, or a list of versions of tools used by specific attackers. In SOC, the threat intelligence management utilizes the information:

Definition 4.4: Threat intelligence management

The "collection, analysis, production, and sharing of cyber threat intelligence." [HBH23]

²³ tactics, techniques, and procedures (cf. glossary, page 329)

CTI is exchanged in different formats [RSK20; Ran+20], such as proprietary JSON²⁴ formats, STIX²⁵ [JPD21], or CSAF²⁶ [RHS22] for exchanging vulnerability information. The CAPEC²⁷ and ATT&CK® catalogs are offered in STIX format, for example. For the automotive domain, a research project defined the openXSAM format [ite23].

a) Closed communities To securely exchange threat information, closed forums such as ISACs and technical platforms exist, e.g. MISP^{28,29}. Closed access is essential to maintain the confidentiality of knowledge about attackers and vulnerabilities. This also allows the exchange of very detailed information, such as IP addresses of attackers. ISACs are established in various sectors³⁰ and countries³¹.

Definition 4.5: Information Sharing and Analysis Center

"Information Sharing and Analysis Centres (ISACs) are non-profit organizations that provide a central resource for gathering information on cyber threats (in many cases to critical infrastructure) as well as allow two-way sharing of information between the private and the public sector." [Eur18]

In the automotive field, to date, two ISACs are established: the Auto-ISAC, and the J-Auto-ISAC³² established particularly for Japan in 2021³³. The aim of Auto-ISAC³⁴ is to provide governmental institutions, vehicle manufacturers and suppliers with a secure platform for exchanging information on vehicle security, such as known attacks or new vulnerabilities, in order to increase the cybersecurity

²⁴ JavaScript Object Notation

²⁵ Structured Threat Information eXpression

²⁶ Common Security Advisory Framework

²⁷ Common Attack Pattern Enumeration and Classification™ (cf. glossary, page 324)

²⁸ Malware Information Sharing Platform

²⁹ <https://www.misp-project.org/>

³⁰ e.g., financial sector <https://www.fsisac.com/>

³¹ e.g., Luxembourg <https://cert.lu/>

³² Japan Automotive Information Sharing and Analysis Center

³³ <https://j-auto-isac.or.jp/>, accessed on Apr. 03, 2024

³⁴ <https://automotiveisac.com/>

of vehicles globally. Developed on the initiative of NHTSA³⁵ [McC+14], the Auto-ISAC commenced operations in 2016 [Adm22].

b) Public and open sources Besides closed platforms, public scientific [SDK19] and commercial [Ups24a] attack databases and non-profit organizations exist. For example, the ASRG³⁶ is a community-driven initiative that promotes knowledge-sharing and networking opportunities in the field of automotive security research, driven by similar goals across the industry.

Moreover, TTP-based (cf. Definition 4.6) threat models and other threat lists are public CTI. For example, OWASP³⁷ maintains top-ten lists of threats to web applications³⁸ and Kubernetes³⁹. To make use of public sources relevant threats of the model need to be selected, for example based on the assets actually available and their criticality. MITRE ATT&CK® [Str+20] is a matrix-based threat catalog. It consists of tactics (phases) that categorize steps in the course of an attack. MITRE currently maintains ATT&CK® matrices for three domains: enterprise, industrial control systems and mobile. Each matrix models attacker behavior as a TTP model.

³⁵ National Highway Traffic Safety Administration

³⁶ Automotive Security Research Group

³⁷ Open Web Application Security Project

³⁸ <https://owasp.org/www-project-top-ten/>, accessed on Jan. 09, 2025

³⁹ <https://owasp.org/www-project-kubernetes-top-ten/>, accessed on Jan. 09, 2025

Definition 4.6: Tactics - Techniques - Procedures (TTP)

TTP form a hierarchy, that describe "the behavior of an actor. A tactic is the highest-level description of this behavior, while techniques give a more detailed description of behavior in the context of a tactic, and procedures an even lower-level, highly detailed description in the context of a technique" [NIS16]. "Tactics describe the strategic objectives or reasons behind an attacker's actions. Techniques outline the methods they employ to achieve these objectives, while Procedures specify the exact sequence of actions taken." [Yek+23]

Each tactic corresponds to a step in a multi-stage cyber attack. ATT&CK®'s tactics have evolved from the cyber kill chain, an attack lifecycle model developed for the military [Clo09]. However, the order of the tactics is not necessarily sequential, as real attackers may skip, combine, or repeat steps [KM22]. Techniques and sub-techniques of a tactic contain a technical description, potential indicators for detection, possible remedies, and references to real-world examples of the threat (procedures).

Several TTP-based threat catalogs were developed for the automotive domain. The ASRG Threat Catalog [Gro23] maps all threats to STRIDE (ThreatClasses view) and categorizes them into 13 attack tactics (matrix view). The ATM⁴⁰ [SC24] of the Auto-ISAC (cf. Section *External sources* on page 80) uses the same tactics, but references examples for the techniques. The VATT&EK⁴¹ approach [Yek+23] models 14 tactics tailored to the automotive and railway sectors. The associated techniques extend the MITRE ATT&CK®, Auto-ISAC, and ASRG matrices. Another matrix is being promoted commercially by VicOne [Vic24], without publicly providing details about the techniques. VicOne's approach was mapped to UN R155's generic threat list [Ani23]. Besides discussing the necessity of tool support for the automation of attack path analysis, the benefits of a catalog as a

⁴⁰ Automotive Threat Matrix

⁴¹ Vehicle Adversarial Tactics, Techniques, and Expert Knowledge

guideline are highlighted. Yet, regular catalog updates would be required in case of new threats.

MITRE's CAPEC is a hierarchical model of common challenges attackers experience when exploiting weaknesses. In its taxonomy, an attack pattern is the destructive counterpart of a design pattern. A pattern informs about the design and execution of an attack and potential mitigations. Besides threat modeling [OK19], CAPEC was used in the automotive field to map UN R155 threats [Som+24a].

c) Further external sources Closed communities and open sources mainly provide information about the general approach of attackers or about other companies that are currently under attack. What is relevant for a particular company must be analyzed. Moreover, for public sources such as websites, the correctness of the information must be checked [ISO20a].

In contrast, bug bounty programs are more targeted, as they directly affect the manufacturer of the vehicle.

Remark 4.2: Bug bounty program

A bug bounty program is "a method of compensating individuals for reporting software errors, flaws, or faults ("bugs") that might allow for security exploitation or vulnerabilities." (NIST SP 800-216, [NIS19])

Nevertheless, vulnerabilities found by white hats that are reported to the manufacturer via such program must be checked for correctness and relevance. A defined interface is necessary [BK21], which is implemented, for example, by GM, BMW, Mercedes-Benz, Bosch, Continental, and Siemens. Information from suppliers is also of direct importance, as vulnerabilities in installed supplier components are immediate weaknesses in the field. Whether these are exploitable vulnerabilities is the subject of vulnerability handling. The exchange of information with suppliers can, for example, be defined via the cybersecurity interface agreement [KA21].

Information from the dark web involves, for example, previously unknown vulnerabilities and malicious code to exploit them, which are sold by criminals. Therefore, these sources may contain information on various manufacturers and suppliers. Though it may be very useful [Gol+21], searching for relevant information requires preparation, such as filtering.

Internal sources

While CTI focuses on knowledge from external sources, internal sources can also contribute significantly to an understanding of the cybersecurity situation. This also includes external sources, such as vulnerability databases, that are linked with internal information.

a) Vulnerability databases Vulnerability scanning, though ISO/SAE 21434 lists it as "information from the field", connects external knowledge with internal sources, as public databases are matched against the own assets.

Remark 4.3: Vulnerability scanning

Vulnerability scanning uses tools (hardware and/or software) to identify hosts, software, and associated vulnerabilities (CVEs, CWEs, and others) (based on NISTIR 8011 Vol. 4, [NIS19]).

Depending on the tool, a SBOM (cf. Remark 4.4) can be generated. Generating an SBOM and matching against vulnerability databases may also be termed SCA⁴².

⁴² software composition analysis

Remark 4.4: Software Bill of Materials

"A formal record containing the details and supply chain relationships of various components used in building software. Software developers and vendors often create products by assembling existing open source and commercial software components. The SBOM enumerates these components in a product." (from NIST SP 800-161r1, [NIS19])

The identification of all software used, including in supplied binaries, especially open source, is the prerequisite for being able to compare against known vulnerabilities. Accordingly, knowledge of the software is an important contribution to monitoring [Ken21].

b) Source code and other development artifacts Vulnerability scanning closely relates to other code analysis techniques. Some scanning tools can detect sensitive information (such as passwords) and misconfigurations in addition to CVE vulnerabilities. Testing techniques, especially dynamic testing that requires software to be executed, can find similar weaknesses. Static analysis (scanning only the source code) can identify common problems such as certain buffer overflows. Fuzzing (random or erroneous input data) and penetration testing (testing for known attacks) are other common development practices that can be repeated during production to find weaknesses. In addition, the required control of changes to artifacts through a change management process [ISO21] could provide information about unauthorized changes. One challenge is to know what software is actually running in the vehicle, as updates are not necessarily visible to the manufacturer [Ken21]. Finally, TARA is also used to identify vulnerabilities. Within Motional's process [Wil23], for example, vulnerability management is already applied during development, following TARA.

c) Field returns and quality reports In contrast to other internal sources, an infrastructure for processing reports from the field is established, which is available to all customers via dealers and support contact points (cf. Remark 4.1). The

staff responsible for safety and quality forwards any relevant problem reports to the technical departments, which address any necessary changes. The literature suggests that such support organizations should be maintained and that security-related problems should be passed on to the cybersecurity operations personnel [HG18].

d) Offboard data The statistics (cf. Figure 2.15) do not describe which information is part of offboard data. From the literature, some relevant internal sources besides the aforementioned categories were identified. First, offboard data includes incidents from the vehicle ecosystem, e.g. from the IT of the OEM or telecommunications service providers [Olt18; May+24]. Besides, honeypots are an offboard source, which was first proposed in the automotive sector in 2008 [Ver+08]. Deliberately providing a vulnerable system, attackers think they are attacking a real system. Hence, honeypots are a way of collecting information about attacks and attackers during their actions. Finally, ISO/SAE 21434 suggests taking past analyses into account. Although this is not taken up by automotive-focused literature, correlation with past attacks is established in IT cybersecurity (cf. Section *Enabling comprehension of the collected information* on page 90).

4.2.3 Cybersecurity information management

Besides the generation or acquisition of information from various sources, cybersecurity monitoring includes (pre-)processing of information (cf. Definition 1.7), for which specific tools are used.

Definition 4.7: Cybersecurity information management

(Cyber)security information management systems are applications that gather or receive, normalize and store cybersecurity information.

With the definition of monitoring chosen in this thesis (including pre-processing, cf. Definition 1.7), information management constitutes a part of it. Procedurally, information management is therefore required for monitoring, but technologically

these are separate capabilities of the SOC (cf. Section *Capabilities* on page 64). The addition of integration and inspection or exploration capabilities to CIM⁴³ systems results in the capabilities of a SIEM system.

Definition 4.8: SIEM

A SIEM is defined as "Application that provides the ability to gather security data from information system components and present that data as actionable information via a single interface." [ST11]

SIEM systems combine CIM with analysis methods, e.g. rules to raise an alarm, and interactive interfaces (cf. Figure 4.8). T-Systems, for example, calls SIEMs "the SOC's alarm system"⁴⁴. Applied in the automotive context as the VSOC's central tool (cf. Figure 4.3), T-Systems SIEM rules are based on attack scenarios represented with a kill chain-based structure, similar to an attack catalog (cf. Section *Public and open sources* on page 82).

Storage in CIM and SIEM systems usually involves databases [Mil11]. Collection can be either a push or a pull process [Mil11], i.e. CMS send their information to the CIM or SIEM system, or the CIM / SIEM system requests the data. Normalization involves *parsers*, that transform from one data format to another [FCM21]. If CSI is produced in different formats, normalization yields a homogeneous set of information.

Normalization is the prerequisite for integration of data from several sources [FCM21]. Examples for the combination of several data sources are correlation⁴⁵, aggregation⁴⁶ and fusion⁴⁷. The "single interface" for presentation of a SIEM may support capabilities such as pivoting⁴⁸ or view of dashboards.

⁴³ Cybersecurity Information Management

⁴⁴ <https://www.t-systems.com/de/en/security/managed-cyber-defense>, accessed on Jan. 07, 2025

⁴⁵ "Finding relationships between two or more log entries" [NIS06]

⁴⁶ "The consolidation of similar log entries into a single entry containing a count of the number of occurrences of the event. [NIS06]

⁴⁷ "obtain a single stream of data from several disparate sources" [FCM21]

⁴⁸ "ability of going from one data source to another" [FCM21]

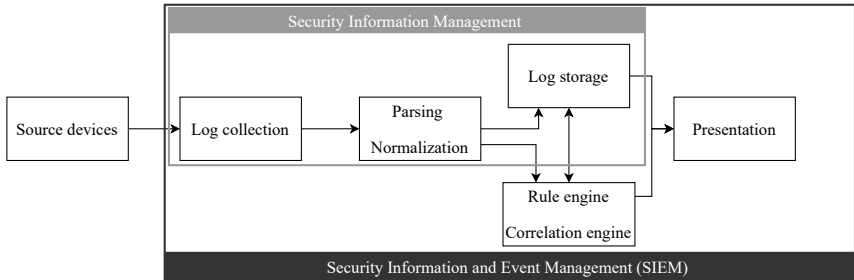


Figure 4.8: Common structure of SIEM systems (derived from [Mil11]).

Protocols and standards

Different standards exist to formalize the information produced by CMS. IDS systems, for example, can usually create custom text formats, syslog, or JSON. However, the most widely supported are IETF⁴⁹ syslog messages [Ger09] (cf. Figure 4.9), in the following called “syslog messages” for short. These plain text messages consist of a header and additional structured key-value data. The priority (part of the header) consists of a *Severity* and *Facility*. The *Severity* ranges from 0 (Emergency) to 7 (Debug), whereas the *Facility* reflects the typical Unix processes (0: kernel - high priority, to 23: local7 - custom process with low priority).

Focusing on the transmission of IDS data, AUTOSAR specified a protocol to forward IDS events, the “Intrusion Detection System Protocol” [AUT20]. Within AUTOSAR also an IdsM⁵⁰ is specified [AUT23b]. Its main purpose is the synchronization, filtering, buffering and forwarding of AUTOSAR *SEv* [AUT24] to a reporting unit⁵¹, which then may send the data to a VSOC⁵³. Filtering involves three possibilities, including sampling (forward every n -th event), aggregation

⁴⁹ Internet Engineering Task Force

⁵⁰ Intrusion detection system Manager

⁵¹ AUTOSAR IdsR⁵²

⁵³ "Depending on the needs of the OEM the data can be propagated to a SOC for further analysis in a SIEM solution." [AUT22]

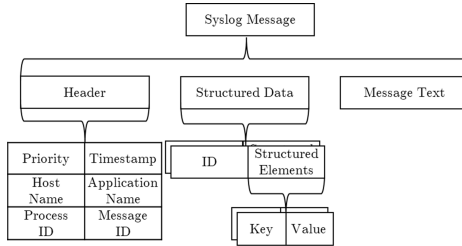


Figure 4.9: Information contained in an IETF syslog message.

(count similar events in time frame) and thresholding (forward after n occurrences). The system can also add a cryptographic signature to provide authenticity of the events.

In the automotive field, for example, syslog was used for the prototypical implementation of a monitoring information management system, that shall observe the safety of software updates in vehicles [KHK21]. Other works collected CAN messages within the vehicle, which are converted to JSON and forwarded via the syslog protocol to an off-the-shelf IT security SIEM system (IBM Security®QRadar®) (cf. Section 4.1.2) [Bar+23b]. In this thesis, the syslog protocol is used to collect data in the vehicle and send it to the VSOC, and to implement a response mechanism (cf. Chapter 9 and Chapter 10) [Gri+23a].

Enabling comprehension of the collected information

A variety of information sources are discussed in the literature on monitoring and VSOCs. However, simply storing data in different silos is not conducive to analyzing the data and identifying vulnerabilities and attacks. Accordingly, it is also a task of information management to link the various data sources in order to grasp the situation. In the automotive environment, data models that link different information are often meta-models for describing development artifacts, on which tools such as PREEvision are based. For cybersecurity, some VSOC approaches suggest using a NoSQL database to store all information

[LSS19; May+24], while others discuss a KG⁵⁴-based approach called Auto-CyberKC [Bar+23a]. In addition, there are approaches for linking architecture, safety and ODD using an ODD-architecture ontology [Erz+22] and automated driving in general with the CoSI-KG [Hal+21; Lue+22]. Another approach is the linking of variant management and dynamic software architectures within a unified variability model [Sch+23].

In IT cybersecurity research machine learning and AI⁵⁵ are often used to comprehend CSI, for example to detect malware or for triage [FC17]. However, analyses in industry also show that trust in these tools is limited [Oes+20]. Another limiting factor for machine learning is the availability of training data, which is particularly the case for the cybersecurity of vehicle fleets. Therefore, this work focuses on the semantically comprehensible structuring and storage of the data does, in contrast to fully automated analyses. This can be achieved by defining an overarching data model across all information sources used, which can serve as a trustworthy and uniform source for analyses [Kom+18], and can also include the in-vehicle data [GS22; Gri+24c]. However, the approach proposed in this work enables several promising directions to benefit from AI in the future (cf. Section 9.4.4).

Structured data storage, in particular KG-based data models, has been researched in various works in IT cybersecurity. CRUSOE [Kom+18] is a KG-based data model, that incorporates seven different information layers: host, system, network, detection and response, access control, mission and threat layer. The KG as a unified data source can show asset criticality and dependencies by fusion of different data sources. CyGraph [Noe+16] defines a layered data model connecting mission dependencies, cyber threats, security posture and network infrastructure. Various data sources are ingested and normalized, including vulnerabilities, network flows, firewall rules and the CAPEC data. SEPSSES considers the relationships between known vulnerabilities in the system (CVE) [Kie+19]. They link CVE to information about the associated weaknesses (CWE), attack patterns

⁵⁴ Knowledge Graph (cf. glossary, page 326)

⁵⁵ Artificial Intelligence

(CAPEC), scoring models (CVSS), and affected components and systems (via CPE⁵⁶). Kurniawan et al. [KEK21] extend this graph with the ATT&CK® matrix data. CoCoa [Onw18] is a data and process ontology for incident handling in SOCs, combining events and logs, network information, structured digital feeds, semi- and un-structured feeds and threat intelligence in a KG. MalRank [Naj+19; NCM21] is a KG for SIEM systems and a ranking algorithm for the maliciousness of nodes (e.g. domain names, IP addresses). The KG builds from proxy, DNS, and DHCP logs collected from the monitored system, and threat information from CTI, including known malicious domains and IPs.

4.2.4 Incident response

Responses in vehicles

Technological ideas to respond to detected cybersecurity problems started already in 2008, addressing visual, acoustic and haptic feedback to the [HKD08a]. However, the main challenge identified in the industry is the ability to respond across fleets. This involves, in particular, the required fast response times, diversity of vehicle types and the scales (number of vehicles) and dynamics (situations, e.g. without network connection), as highlighted in a talk by Mercedes [HC20]. For this reason, it is not only the technologies that are of interest, but also overall response procedures.

Mercedes suggests to implement immediate response actions, such as selectively blocking services, and applying immediate fixes, before performing forensic investigations and develop long-term fixes. Afterward, the "lessons learned" may yield hardening of the vehicles and rolling out of the long term fixes [HYL20]. Immediate responses are also proposed in the scientific literature [LSS19; LSS20] (cf. "ACDC" in Section 4.1.2). The authors' central idea is that updates are unsuitable as the sole response option due to predicted response times of several

⁵⁶ Common Platform Enumeration (cf. glossary, page 324)

weeks. To accelerate this (minutes to days), the authors propose three response levels:

1. Immediate response, containment (minutes): Automated informing of affected vehicles about changed threat level, automated switching of vehicle to safe fallback states preconfigured during development (e.g., disable untrusted services, postpone feature updates, disable services with external communication)
2. Intermediate response, recovery (days): targeted configuration update to deactivate or isolate affected components (e.g., orchestration of services, firewall reconfiguration)
3. Long-term response, eradication (weeks): develop update to fix vulnerabilities and prevent future attacks

From an industrial perspective, the firewall [AUT23a] and IdsM (cf. Section *Protocols and standards* on page 89) specifications in AUTOSAR describe runtime-configurable options. In particular, besides regular updates for adding or modifying rules, the firewall can dynamically activate and deactivate the configured rules at runtime. Furthermore, rules can be configured as state-dependent (e.g., only active in parking mode). For the IdsM, an API allows changing the *reporting mode* via diagnostic access, which determines the application of filters before forwarding data.

Furthermore, related works discuss technical safety measures to reduce or prevent the effects of cybersecurity incidents, such as redundancy and fallback calculations [Ros+20; TW16]. Reducing the physical impacts makes autonomous safety measures an essential aspect of long-term cybersecurity. For incident response, however, measures that can actively and precisely change the attack surface are of greater importance. Accordingly, the focus is on controllable mechanisms to react targeted in unforeseen circumstances.

For example, triggering the existing limp mode can help to reduce the physical impacts of a detected attack [Dag+17; Ros+20]. Another approach is to use orchestration, hypervisors, and containerization to migrate applications to secure

platforms when the previous host is vulnerable or attacked [Häc+23; Ros+20]. After receiving a new service assignment from the VSOC, containerized services were operational again after 1.7s at the latest in a prototypical implementation [Mey+20; Häc+23]. SDN⁵⁷ can be reconfigured to change traffic routing and mitigate or contain ongoing attacks. For example, autonomously triggered by a NIDS, an SDN blocked a malicious flow after a maximum of 328 ms in an experimental automotive network [Mey+20; Häc+23]. Finally, checkpointing and rollback mechanisms can restart applications and restore previous secure states [Ros+20]. Generally, a common approach is a dedicated in-vehicle component that controls response actuation, with response commands transmitted from a backend system [LSS20; Häc+23; LHS22; GS22].

General approaches

IT security standards and guidelines (cf. Table 4.2) supplement the automotive-specific approaches with a generic perspective on response procedures. Overall, three response phases are distinguished: containment, eradication, and recovery.

Definition 4.9: Containment

"Containment provides a reasonable security solution until such time that sufficient information has been collected to address the vulnerabilities and the damage. The containment goals are to prevent an intruder from accessing or exfiltration of data or other information, destroying valuable evidence and tampering with systems while they are being analysed, and using systems to attack other systems, protecting the organizational components from liability." [ISO20a]

⁵⁷ Software-defined Networkings

Definition 4.10: Eradication

"Eradication is the elimination of components of the incident such as malicious code, compromised accounts and passwords, or other compromised systems and information. The goal of eradication is to permanently remove digitally stored data from some form of media either effectively or actually." [ISO20a]

Definition 4.11: Recovery

"Recovery is the restoration of a service, data or system to its normal operational state." [ISO20a]

NIST SP 800-61 [NIS12] emphasizes that containment is necessary for most incidents to gain time to develop a tailored response strategy, but the containment method varies with the incident. The collection of forensic evidence should be taken into account during the response process as well. During the recovery process, an increased *system log level* is recommended to observe occurrences of the same or a similar attack.

Table 4.2: Incident response operation methodology in enterprise IT security standards and guidelines.

Phase	ISO/IEC 27035	Standard
		NIST SP 800-61
Containment	• Blocking (firewall): IP, Port, Mail, proxy, URL / domain • Disconnection (isolation, removal) from local net or internet / public net • Shut down affected or endangered systems • Change routing to eliminate transmissions • Disable user accounts	• shut down system • disconnect system from network • disable functions • redirect the attacker to a sandbox for monitoring the activity
Eradication	• HDD reformat • wiping tools to erase disk • firmware flashing • physical destruction	• Delete malware • Disable breached accounts • Identify and mitigate all exploited vulnerabilities
Recovery	• rebuild from backup • rebuild from scratch • change accounts / passwords • harden against re-occurrence • apply updates • recover to hot / warm site if other site is hardened against incident	• Restore from clean backup • Rebuild from scratch • changing passwords • tighten network security (e.g., firewall rulesets, boundary router access control lists) • install patches • replace compromised files with clean versions

5 Discussion of the current state of automotive cybersecurity operations

5.1 Processes¹

5.1.1 Process(es) for cybersecurity operations activities ①

While introducing VSOCs is proposed from industrial and scientific perspectives, the activities that would be part of a VSOC are unclear. The automotive-focused literature primarily considers the technological realization of VSOCs but does not discuss ISO/SAE 21434's role and suitability for the operations phase. Hence, in the following, IT processes are compared against the automotive standard ISO/SAE 21434. Based on the requirements for the process (cf. Requirement 1-1 to Requirement 1-3), detailed criteria were derived to compare their suitability. Given that UN R155 mandates monitoring, incident handling, vulnerability handling, updating of risk assessments and cybersecurity measures, and timely incident response, the scope of these activities is essential. Further aspects concern the safety-criticality and distributed development, resulting in the following criteria:

¹ Section 5.1 is based in large parts on a publication of this dissertation's author [Gri+23b]: D. Grimm, A. Lautenbach, M. Almgren, T. Olovsson and E. Sax, "Gap Analysis of ISO/SAE 21434 – Improving the Automotive Cybersecurity Engineering Life Cycle," 2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC), Bilbao, Spain, 2023, pp. 1904-1911, doi: 10.1109/ITSC57777.2023.10422100. ©2023 IEEE.

1. Adherence to standard terminology: Security operations for vehicles and IT must be able to cooperate, as attacks on IT systems can impact vehicles. A common terminology is desirable to limit communication challenges between these entities.
2. Overall workflow: All common aspects of cybersecurity operations should be covered. The process should enable swift responses to incidents.
3. Vulnerability and incident handling methodologies: An operations process primarily defines the required phases but should also include practicable options for their realization. In particular, methods for analyzing vulnerabilities and incidents are valuable.
4. Automotive-specific constraints:
 - Clear and binding requirements: Approval organizations assess a manufacturer's process compliance. All assessments should be comparable, which requires unambiguous requirements.
 - Distributed development: Due to distributed development, various companies can cause vulnerabilities. Hence, development and operations require defined communication and responsibilities.
 - Physical safety: Influences and effects on the physical environment must be considered, particularly in the risk assessment.
5. Updating risk assessments: The operational phase must ensure the TARA is up-to-date. Explicit rules for the identification of required updates are essential.

Adherence to standard terminology

To identify process-related gaps in ISO/SAE 21434 concerning vulnerability and incident handling, the focus of the discussion are processes from IT security incident handling and automotive-specific cybersecurity operations processes (cf. Section 4.1). Before comparing the content (steps and workflows), the terminology should be aligned. Even in IT security, there is no commonly agreed upon incident handling terminology, yet there are commonalities which ISO/SAE 21434 deviates from.

Events: A "security event" is often defined as "an occurrence in a system" that may be security-relevant. In contrast, ISO/SAE 21434 defines it as "cybersecurity information that is relevant for an item or component." According to this definition, an event refers to information, whereas the common definition of "an occurrence in a system" is concretely tied to a system state.

Incident: An "incident" is typically defined similarly to "a security event that involves a security violation," whereas ISO/SAE 21434 defines a cybersecurity incident as a "situation in the field that can involve vulnerability exploitation." If "in the field" means "during operations," this definition excludes incidents occurring during development.

Incident response plan: In ISO/SAE 21434 the meaning of the incident response plan is primarily a per-incident document. In contrast, in the NIST standard [NIS12], the response plan is an overarching strategy and "roadmap for implementing the incident response capability," including metrics to measure that capability.

Triage: As used in ISO/SAE 21434, the meaning of triage is inconsistent with the literature. In IT security, the aim is to give the worst incidents the most attention. In ISO/SAE 21434, triage only involves filtering but is not used as a term for prioritization, which would include classification or rating.

Overall workflow

The sequence and steps of the operations process determine how well vulnerability and incident handling capabilities are supported (cf. Requirement 1-1). Firstly, all considered processes [NIS12; Alb+04; ISO16b] describe the need to prepare for vulnerability and incident handling, including preparing necessary information, a step missing in ISO/SAE 21434. Consequently, the aspects to be prepared during the development phase (cf. Requirement 1-3) remain unclear.

Secondly, incident response is iterative in both NIST SP 800-61 and CMU/SEI-2004-TR01. Often, several steps are required to achieve containment first, then

eradication, and finally, recovery of operations. Moreover, a verification of the effectiveness of the response determines if an iteration is required. In contrast, ISO/SAE 21434 does not suggest an iterative process. Response action goals such as eradication and recovery are not part of ISO/SAE 21434.

An incident or vulnerability handling follow-up ("lessons learned") is another concern. In NIST SP 800-61 and ISO/IEC 27035, post-incident activity is one of the main phases, emphasizing its importance. CMU/SEI-2004-TR01 performs post-incident analysis in the sustain and improve phase to derive requirements for process and technology improvements. ISO/SAE 21434 covers learning from experience in [RQ-05-08] as part of cybersecurity culture. However, it is not integrated with the cybersecurity operations processes (cf. Requirement 1-3).

Regarding the time frame and interaction of phases, the vulnerability and incident handling processes in ISO/SAE 21434 focus on the analysis and resolution of vulnerabilities. Incident handling is only invoked as part of vulnerability management, which does not cover all cases. Once cybersecurity monitoring matures, detecting incidents before vulnerabilities may become more common, and in those cases, incident containment should be more critical than finding the underlying vulnerability. Notice that incident handling may require a higher level of urgency, which the literature also emphasizes. As a result, ISO/SAE 21434 only foresees the development of an update or configuration change, which must meet all the requirements of an entire development process. Due to the subsequent connection of incident handling with vulnerability handling foreseen in ISO/SAE 21434 and the required approval of reactions, an adequate reaction speed is not assured (cf. Requirement 1-1).

Vulnerability and incident handling methodologies

The methodological depth of ISO/SAE 21434 differs between vulnerability and incident handling (cf. Requirement 1-1). The TARA methodology forms the basis for vulnerability handling in ISO/SAE 21434. Ultimately, if a vulnerability is found and there is no ready-made solution (e.g., patch by supplier), a risk

treatment decision based on risk values is necessary. Therefore, the inherent problems of TARA remain open for vulnerability handling, such as inconsistent evaluation and cooperation with suppliers.

The methodology for incident response in ISO/SAE 21434 lacks depth and is somewhat ambiguous. Unlike IT standards that emphasize precise actions like containment, eradication, and recovery, ISO/SAE 21434 does not explicitly define these. Instead, it mandates the creation of an incident response plan specific to each incident. Yet, ISO/SAE 21434 also allows the execution of incident response independently of TARA, which contradicts that vulnerability management should determine the remedial actions within this plan.

Automotive-specific constraints: clarity, distributed development, safety

Concerning the automotive-specific requirements for a cybersecurity (operations) process, the considered related works differ from ISO/SAE 21434. While TARA in ISO/SAE 21434 takes physical risks to the environment into account, including usage of TARA within vulnerability handling, other processes do not consider safety (cf. Requirement 1-2). Besides discussing IT cybersecurity processes, integrating SOTIF monitoring with cybersecurity incident monitoring is an interesting idea regarding safety, as safety-relevant situations are recorded in particular [Pia22].

A positive aspect of ISO/SAE 21434 is the binding nature of the requirements and work products, which makes it a suitable assessment basis for approval organizations. The number of publications aiming to change TARA shows a lack of clarity in ISO/SAE 21434's calculations. As a result, inconsistent assessments in TARA also impact operations, especially vulnerability handling. As discussed regarding the workflow, the amount of requirements in the operations area is generally insufficient. While it does not feature a concrete list of requirements and work products, CMU/SEI-2004-TR01 is clearly described, including the required inputs and outputs of steps. In contrast, NIST SP 800-61 and ISO/IEC

27035 are vaguely formulated and do not involve binding assessment criteria. ISO/IEC 27035 also has an unclear document structure.

Concerning distributed development ISO/SAE 21434 could be improved (cf. following paragraph). However, cybersecurity interface agreements can provide a framework to formalize cooperation. Nevertheless, its use during operations to exchange information about vulnerabilities and incidents is still the subject of research. In the other standards, cooperating with suppliers is not addressed in detail. ISO/IEC 27035 discusses the role of information exchange with suppliers.

Updating risk assessments

ISO/SAE 21434 leaves the TARA methodology to the standard's adopter. As experience with TARA increases, common problems have become apparent, such as inconsistent evaluation and the demanding organization of risk assessment across the ecosystem of the industry [KA21; Gre+22]. While inconsistent evaluation can lead to significant misjudgments of risks, particularly during the development phase, managing risk assessment along the supply chain also has a major impact during the operational phase.

UN R155 mandates a process "for ensuring that the risk assessment is kept current" [EEC21a]. ISO/SAE 21434 applies the TARA during the concept phase and notes that the event evaluation phase can introduce updates to threat scenarios. Accordingly, there is a gap in the standard, not to consider when the other artifacts of TARA are to be updated (cf. Requirement 1-3, Requirement 1-1). Furthermore, implementing TARA updates remains uncertain, including the specific information that may be required. Such an update could be necessary if a supplier identifies a new vulnerability within its subsystem. In this case, the attack feasibility on dependent components could increase. How can such knowledge be shared with the manufacturer? ISO/SAE 21434 does not provide any information on this. Moreover, if information is shared, how can their compatibility be ensured?

Automation approaches (cf. Section *ISO/SAE 21434 in practice* on page 69) improve the consistency of created TARA. However, their use for updating the TARA during operation has not been under research. Guidance on what to share between customer and supplier during development partially addresses the problem [KA21], but doesn't address updates of TARA during operation. The literature considers maintenance responsibility for TARA as part of the CSMS or VSOC [Lie21], including changes in the security posture. Overall, further guidance in applying TARA and keeping it up-to-date is required.

Of the other standards, only CMU-SEI-2004-TR01 [Alb+04] considers risk assessment of the current infrastructure (i.e., architecture). In addition, as incident response outcomes are used in the prepare and sustain phase, updates to risk assessments may be identified. However, updating is not as formalized as in the automotive field, and keeping risk assessments up-to-date is not the primary focus. The other processes mainly focus on incident handling and do not consider assessing risks.

5.1.2 Procedure to specify monitoring requirements ②

With the introduction of ISO/SAE 21434, cybersecurity requirement identification using risk assessments has become established. TARA is *the* approach to define high-level (functional architecture) and detailed requirements. While several publications discuss TARA automation, practical application, and improvements, a crucial problem is that requirements for cybersecurity operations may be overlooked, particularly for monitoring and response. In general, there is no obligation to specify explicit monitoring requirements (cf. Requirement 2-1). Indeed, ISO/SAE 21434 addresses monitoring and considers several data sources. Besides, it mentions that "Cybersecurity claims can be considered for cybersecurity monitoring" and that the description of cybersecurity controls can include "conditions for achieving cybersecurity goals, e.g. prevention of the compromise, detection and monitoring of the compromise." Yet, neither the standard nor related work discusses how to derive specific monitoring requirements with due consideration of the risks (cf. Requirement 2-2). About the technological

implementation, it is also important to note that CMSs have different vantages, and its monitoring domain (network, host, service) may limit which attacks a sensor may observe.

While applying probabilistic approaches from IT security for the placement of CMS is possible, it would require estimates for probabilities or traces of attacks, which are not available, especially for new developments. Also, the interaction of probabilistic methods with risk assessments is insufficient, which could lead to poor alignment with the risks identified during TARA. A methodical approach such as the ACF (cf. Section 4.1.4) is more suitable, as a connection to risk assessment is feasible. In particular, the first two steps of the ACF, resulting in quantified risk levels, are similar to TARA. A further finding is that although TTP models are gaining importance in the automotive sector (cf. Section *Public and open sources* on page 82), in contrast to the original ATT&CK® matrix, no publication describes possible detection sources, which limits the usability of the automotive-specific TTP models to information sharing.

5.2 Technologies

5.2.1 In-vehicle monitoring of cyber and physical aspects ③

Network security monitoring

While publications combine several data sources in the vehicle, e.g., [Val+24], how to monitor the whole in-vehicle network best is not discussed (cf. Requirement 3-1). Using Collins' taxonomy of CMS (vantage, domain, action), the analysis of the current research shows a focus on the network domain while the industry is also pushing the host domain. A good technological status can be assumed for these two domains, even if several established IT monitoring technologies are not used yet in the vehicle sector. These include flow and packet collection, for example. The predominantly static network configurations, which

are a specific characteristic compared to IT networks, are considered in particular by AUTOSAR's SEvs. In contrast, the service domain is not the focus of research and industry, although DLT could serve to collect service-level data. However, monitoring the service domain is essential, especially for SDV (cf. Requirement 3-3), where applications are becoming more dynamic due to containerization and orchestration.

Besides Collins' CMS domains, IT cybersecurity monitoring involves further technical sources (cf. data sources in MITRE ATT&CK®²). In particular, proxy servers, DHCP and DNS logs (e.g., [NCM21]), container and VM management logs, and the configuration of network devices and hosts may be available in SDVs (cf. Requirement 3-3). Moreover, monitoring does not involve typical automotive-specific services, such as update management and network management. As a result, while the established domains are technologically covered (host, network) or should be improved (service), the services required at the system level are underexplored.

Physical situation monitoring

Currently, the literature does not systematically consider the physical situation of vehicles for cybersecurity (cf. Requirement 3-2) besides detecting anomalous signal values. Although safety is an impact assessment criterion in the TARA, efforts to monitor possible impacts are limited. For instance, Piątek links safety monitoring with security monitoring by monitoring predefined triggers [Pią22]. Additionally, ISO/SAE 21434 recommends that TARA should include defining the operational context of identified risks. Publications include, for example, the status of ECUs [Olt18]. As the ODC define the relevant status and environment attributes for the safety of a specific (driving) function, the standards ISO 34503 and 34504 [ISO23a; ISO24b] have laid the ground for a standardized taxonomy of relevant physical aspects. As a result, the operational context includes at least the environmental conditions, scenery, dynamic elements, and the vehicle's status.

² <https://attack.mitre.org/datasources/>, accessed Jan. 08, 2025

5.2.2 Fleet-wide collection ④

The management of SEVs in AUTOSAR shows the industrial interest for in-vehicle CIM, providing filtering and aggregation capabilities similar to IT tools (cf. Section *Protocols and standards* on page 89). These mechanisms can be the basis for sending only currently relevant information to a backend (cf. Requirement 4-1). Scientific works remain superficial in this field. For example, packing all in-vehicle network data into JSON messages and sending them to a SIEM does not seem to be a sensible concept due to the large traffic volumes in the vehicle. Centralized in-vehicle management components with some processing [AUT23b; LSS19; LHS22] seem to be a better approach to enable collecting information from various CMS components.

Regarding protocols, syslog is an established standard with fields for source, criticality, timestamp, and free and structured text. The AUTOSAR IDS protocol contains similar information but could be transmitted via CAN and may be digitally signed to ensure the authenticity (cf. Requirement 4-2). However, encryption and authentication of the receiving side are not built-in. In contrast, IT CIM tools commonly feature TLS-based encryption and authentication. Overall, while in-vehicle data can be represented well both with IT or AUTOSAR protocols, securing the transmission to a VSOC is not covered within AUTOSAR (cf. Requirement 4-2).

5.2.3 Connection with additional information ⑤

IT security standards propose both NSM and CTI data sources, covering the technical and strategic perspective of CSA. In comparison, ISO/SAE 21434 focuses on external data sources, mostly related to CTI. For example, IDS are not explicitly considered, which are the focus of research in automotive cybersecurity technologies. Publications in the automotive field consider external and OEM-internal sources, but linking the information to in-vehicle data is not discussed in detail [Ken21; LSS19]. One work connects vehicle-internal data with an attack taxonomy and past attacks [Bar+23a], and another implements several VSOC

components without focusing on data semantics [May+24]. Consequently, the relevance and availability of additional sources and their combination with the vehicle's internal data remains open.

External sources

Overall, the external data sources recommended in ISO/SAE 21434, such as researchers, organizations, and commercial sources, are established in IT cybersecurity and gaining relevance in the automotive industry. Accordingly, all sources should be considered (cf. Requirement 5-2). For example, publications discuss the integration of CTI in VSOC [Olt18; May+24; Bar+23b], and the provision of aligned CTI collections in different formats (cf. Section *Public and open sources* on page 82), with a focus on MITRE ATT&CK® matrix variants. ISACs are important sources, as it is the second most used external data source in the vehicle industry alongside (public) vulnerability databases in 2023. The increasing number of automotive-specific CVE entries [Vic23] shows the acceptance of public vulnerability databases, which manifests in being the number one source for monitoring in 2024. The implementation of bug bounty programs by many manufacturers shows their importance.

Internal sources

According to statistics, development data, such as TARA artifacts, are used by more than a third of the industry participants surveyed (cf. Figure 2.15). However, automotive-specific research (cf. Requirement 5-1) does not consider these sources. Generally, IT cybersecurity monitoring does not incorporate risk analysis results because risk assessment is not mandatory in this field. However, processes such as NIST SP 800-61 [NIS12] and CSA data models (cf. Section *Enabling comprehension of the collected information* on page 90) discuss to prepare other (internal) data sources for monitoring and incident handling. These focus on the documentation of systems, including the relationships between hosts and software,

(business) missions of the components and their criticality, and the cybersecurity controls used. Automotive publications do not consider these aspects for VSOCs.

Database technologies

Long-term storage of connected fleet, OEM-internal, and external data (cf. Requirement 5-3) requires a database. Various database technologies are in industrial use today, including relational databases, Knowledge Graphs, document stores, and time-series databases. However, the suitability of these technologies for use in VSOCs depends on several critical aspects (cf. Requirement 5-4 and Requirement 5-5). In particular, the specific use cases in a cybersecurity operations process (e.g., escalation from triage to incident handling), and the characteristics of cybersecurity information, threat intelligence (often incomplete, with uncertainty), and fleet data (distributed worldwide, a high number of parallel information flows) determine the evaluation criteria. As a result, the comparison of the database technologies (cf. Table 5.1) considers how each database technology handles schema flexibility, write performance, read performance, query language standardization, maturity, and scalability. These are vital factors to support the operational and analytical requirements of VSOCs that deal with varied and dynamic cybersecurity data.

Overall, the Knowledge Graph has several benefits for the considered use case. For instance, the comparison between MySQL and neo4j as typical implementations of relational (SQL³-based) and Knowledge Graph models showed the strengths of the graph-based approach when querying large data sets across multiple relationships (1.5 million entities and > 5 relationship steps). In contrast, MySQL was faster when loading (writing) large data sets [GH24]. However, in the given use case, millions of parallel but small-sized writes are more important. Particularly for graph databases, a recent study [Bes+23] does not indicate

³ Structured Query Language

a favored implementation (e.g., LPG⁴, RDF⁵). Depending on the application, different memory-storage implementations behind the graph-based query languages can be better or worse. Ultimately, evaluating different graph database implementations with the actual data would be required.

Table 5.1: Comparison of database technologies. Last row sums up points per criterion: o: 0 pt., +: 1 Pt, ++: 2 Pt.

	Relational database	Knowledge Graph	Document-based database	Time-series database
Schema flexibility	o	++	+	o
Write performance (bulk)	++	o	+	+
Write performance (frequent, small)	o	++	++	++
Read performance (multi-relations)	+	++	+	+
Query language standardization	++	+	o	o
Maturity (access control, storage)	++	+	+	+
Scaling, distribution	o	++	++	++
Sum	7	11	8	7

a) Schema Flexibility Schema flexibility addresses the ability of the database to handle evolving data structures and inhomogeneous data. Relational databases are typically rigid, requiring predefined schemas that can be cumbersome to modify. Knowledge Graphs, by contrast, are highly flexible, allowing for dynamic relationships between entities. Inhomogeneous relationships, for example, linking some observables to software components and others to a specific host or network address, are supported in Knowledge Graphs natively. Inhomogeneity also applies to properties, such that nodes and edges can have individual properties, even if they belong to the same type as another node or edge.

⁴ Labeled Property Graph (cf. glossary, page 327)

⁵ Resource Description Framework (cf. glossary, page 328)

Relational databases would have to create several tables to store the links and properties for each associated link or property. Document-based databases offer moderate flexibility, supporting semi-structured data but lacking the entire dynamicity of graph models. Time-series databases are also relatively rigid in schema design, focusing primarily on time-indexed data rather than accommodating diverse or dynamic schemas.

b) Write Performance (Bulk) Bulk write performance is important when inserting large amounts of data at once. Relational databases are optimized for batch operations, benefiting from transactional integrity. Knowledge Graphs, while efficient for small updates, do not handle bulk writes as well due to the maintenance of relationships across entities. Document-based and time-series databases are slightly more efficient for bulk inserts due to their focus on high-volume data with less schema flexibility.

c) Write Performance (Frequent, Small) Frequent, small writes are common where data is constantly being updated or appended, such as data from a vehicle fleet. Knowledge Graphs, document-based and time-series databases excel in this scenario due to their optimization for high-velocity data. Relational databases struggle more with frequent small writes due to the overhead of maintaining transactional consistency and relational integrity.

d) Read Performance (Multi-Relations) The ability to efficiently query multiple relationships is essential in VSOCs, where such queries are needed to correlate multiple information. Knowledge Graphs are optimized for this purpose, providing fast traversal capabilities that allow for efficient querying of interconnected data. Relational databases also handle multi-relational queries, though join operations limit the performance at scale. Document-based databases are less efficient at handling such queries across multiple documents, particularly when relationships span multiple entities. Time-series databases are typically only optimized for time-based aggregations.

e) Query Language Standardization Standardization of query languages is helpful for ease of use and interoperability. Relational databases use SQL, a well-established language with a high degree of standardization across implementations. Knowledge Graphs often employ specialized query languages like Cypher®, which are not as widely standardized. However, the recent standardization of GQL⁶ in ISO/IEC 39075 [ISO24a] could soon overcome this challenge. Document-based and time-series databases typically rely on proprietary query languages (e.g., MongoDB’s MongoQL), which are flexible but lack the broader standardization of SQL.

f) Maturity (Access Control, Storage) Maturity in terms of access control and data storage is vital for ensuring data security and integrity in multi-user environments. Relational databases are highly mature, offering robust access control mechanisms, such as role-based access control and fine-grained access policies. Knowledge Graphs, document-based and time-series databases are less established in terms of access control and fine-grained data management, and their flexibility may complicate controlling security.

g) Scaling and distribution Scalability is important when dealing with large, growing datasets. Knowledge Graphs and document-based databases excel in horizontal scaling, allowing them to distribute data across several database instances and handle increasing workloads effectively. Time-series databases are also built with scalability in mind, designed to handle large datasets of time-ordered data, making them ideal for environments with high data ingestion rates. Relational databases, on the other hand, typically scale vertically, and while some can be horizontally sharded, this requires significant efforts and is often less efficient compared to the other database types.

⁶ Graph Query Language

Data models for semantic connection

The semantic connection (cf. Requirement 5-4, Requirement 5-5) of monitoring data has not yet been sufficiently addressed in the automotive sector. One work links SIEM alerts with an attack database, providing a contribution in this direction [Bar+23a]. Besides, NoSQL databases without in-depth semantics are used in other publications [May+24; LSS19]. In contrast, IT cybersecurity research provides various data models. These approaches have the graph-based database modeling in common, which is also shared by the automotive work [Bar+23a; Bar+23b].

To discuss the differences between IT cybersecurity data models and information usage in automotive publications, the obtainable information aspects per monitoring source (cf. Section 4.2.2) were analyzed. Note, that the *offboard data* source was differentiated further. The aspects to be obtained from the vehicle, internal and external sources were classified into four views (cf. Table 5.2). Each of these views is identified in the rest of this dissertation with the color introduced here. This categorization covers all information that is considered in the automotive field and/or part of IT data models. Among the selected views, there is no dedicated one on identity management and access control as in CRUSOE [Kom+18] (cf. Table 5.3), since to date, user accounts do not play a comparable role in vehicles as they do in IT. The aspects relating to the mission of the company or system were also not considered a separate view [Noe+16; Kom+18]. In contrast, this thesis considers a vehicle's mission to be the provision of the customer features, which includes, in particular, the vehicle's long-term driving task.

As a result, one-to-one reuse of the IT cybersecurity data models is not possible, as they do not fully cover the vehicle specifics. While NSM, (IT) network architecture, CTI and external data sources and partly the incident handling process are covered, other aspects are missing. These stem from the fundamental differences between vehicles and IT networks that continue to exist for SDV, namely the number and variance of assets, OSs, the lifespan of the vehicles, the operational context, model-based development processes, and mandatory risk assessments. Considering the number and variance of assets, variant management

Table 5.2: Information obtained from data sources for monitoring, origin of data source (V: in-vehicle, I: OEM-internal, E: external), and classification in four views, marked by colors.

Source	Origin	Information	View	Color
Network security monitoring	V	observables	Vehicle shadow	
Physical situation monitoring	V	environment, scenery, dynamic elements, status	Vehicle shadow	
Closed communities, deep web	E	threat actors, techniques, tools, indicators	Threat information	
Open sources	E	threat actors, techniques, tools	Threat information	
Bug bounty, responsible disclosure	E	reports	Analysis and handling	
Vendors, suppliers	E	weaknesses, events reports	Analysis and handling Analysis and handling	 
Dark web	E	tools, procedures, indicators reports	Threat information Analysis and handling	 
Vulnerability databases	I/E	weaknesses	Analysis and handling	
		reports	Analysis and handling	
		CVE reference	Threat information	
		versions, software components	Development	
Source code scanning	I	weaknesses	Analysis and handling	
Development artifacts	I	Features, configurations	Development	
		threat scenarios, attack paths, damage scenarios, requirements, cybersecurity measures	Development	
		components, capabilities, dependencies, versions	Development	
		networks, hosts, software	Vehicle shadow	
Field returns, quality reports	I	reports	Analysis and handling	
offboard data (ecosystem)	I	reports	Analysis and handling	
offboard data (honeypots)		reports, observables	Analysis and handling	
		threat actors, tools, procedures, indicators	Threat information	
offboard data (past analyses)		incidents, events, vulnerabilities, weaknesses, courses of action	Analysis and handling	

information should be integrated, which is not the case in the available data models.

5.2.4 Immediate response over the air ⑥

Reaction capability has been taken up by industry and research alike. Starting with the first user-centric reactions, research has developed to software-driven adaptability. For example, AUTOSAR defines the interfaces for adaptation of security measures at runtime (cf. Requirement 6-2). Langer et al. [LSS19; LSS20] addressed containment, eradication and recovery and related these steps to the vehicle fleet, showing that rapid adaptation of the architecture (cf. Requirement 6-2) is possible. In contrast, reactions proposed in ISO/SAE 21434 (development of updates) is little compared to IT standards (cf. Section 5.1.1 and Section *General approaches* on page 94).

A few works consider automation of responses in the vehicle (cf. Requirement 6-2), but adapting monitoring was not discussed. In IT, for example, a customizable system log level is recommended. ISO/SAE 21434 also recommends defining procedures for collecting additional monitoring data, but the realization is up to the implementer. Furthermore, securely enabling responses (cf. Requirement 6-1) is not addressed beyond using TLS [LSS19].

5.3 Summary

5.3.1 Processes

UN R155 mandates monitoring, vulnerability and incident handling, and response capabilities (cf. Chapter 1). While the automotive industry familiarizes itself with these processes, especially with ISO/SAE 21434, gaps become apparent when compared to the experience from the more mature IT security industry. Established processes (cf. Section 4.1.3) indicate potential improvements for ISO/SAE

21434, including terminology, incident handling, and regarding the overall process scope and workflow. These shortcomings are evident in comparison to all standards considered, while ISO/SAE 21434 scores well when it comes to bindingness, distributed development, considering physical impacts on safety, and vulnerability handling (cf. Figure 5.1). The long-term handling of risk assessments, particularly the necessity of updating them, has been identified as a gap in all processes. Generally, a new operations process for continuous cybersecurity of vehicles is required.

Currently, TARA is the focus of development according to ISO/SAE 21434, achieving risk-based prioritization of cybersecurity controls. While this yields a good preventative scope, there is no procedure to specify monitoring requirements in ISO/SAE 21434. However, TARA covers half of the ACF, a structured manual approach to identify what information should be monitored, which may be a starting point for improvements.

5.3.2 Technologies

While various CMS and some reaction options are discussed in industry and science, for both aspects an overarching structure or approach is missing. For NSM, the service domain is not sufficiently covered by CMS, and the physical situation is not considered. Also, the current CMS domain classification does not sufficiently cover common services for the vehicle's network functionality. Besides, the ability to adapt monitoring is lacking.

Management and transfer of CSI from the vehicle to the cloud is addressed in industry. However, sending reactions back to the vehicle is not established. Overall, management of monitoring information and reactions in the vehicle is rather a problem of integration with the heterogeneous E/E architecture. However, linking vehicle-internal information with additional knowledge in the backend has significant potential for improvement. CSA has not been addressed for the decisions of the VSOC, but data models from IT could be suitable with some adjustments (cf. Table 5.3). For example, while MBSE formalizes the configuration, logical

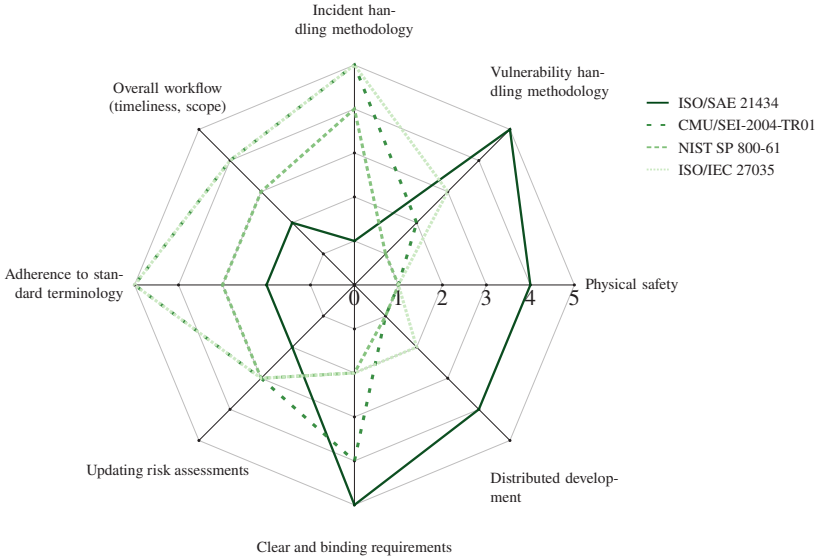


Figure 5.1: Qualitative comparison of ISO/SAE 21434, CMU/SEI-2004-TR01 [Alb+04], NIST SP 800-61 [NIS12], and ISO/IEC 27035 [ISO16b; ISO20a] as an overall cybersecurity operations process. 0: bad / immature capability, 5: good / mature capability.

and E/E architecture, this expert knowledge is not formally integrated yet with in-vehicle data, although both data sources are used in practice. In contrast to MITRE ATT&CK®, automotive TTP models currently lack a description of data sources for detecting the techniques. Operational context is mostly discussed in relation to automated driving.

Table 5.3: Comparison of the covered information of different data models. ○: partially covered, ●: well-covered

Data model	Field of research	Contained information								
		Variant management	Logical architecture	E/E architecture	Risk management	Operational context	Cybersecurity information	Analysis and handling	Threat information	Identity and access control
CyGraph [Noe+16]	CSA			○		○	○	○	●	
CRUSOE [Kom+18]	CSA		●	○	○	○	●	●	○	●
CoCoa [Onw18]	CSA			○			●	○	●	
SEPSES [Kie+19; KEK21]	CSA						●	○	●	
MITRE ATT&CK® [Str+20]	CTI						●		●	
Automotive TTP models [Yek+23; Vic24; SC24]	CTI								●	
Auto-CyberKC [Bar+23a]	VSOC						●		●	
Unified variability model [Sch+23]	Variant management	●	●	●						
PREEvision [Vec24]	E/E architecture	●	●	●						
ODD-architecture ontology [Erz+22]	ODD & ADAS		○			●				
CoSI-KG [Hal+21]	Driving situation					○				

6 Conception of a cybersecurity operations approach for vehicles

Various tasks are associated with cybersecurity operations of vehicles (cf. Task T1 to Task T5 on page 12), which are legally binding under UN Regulation 155. This dissertation explores them regarding the required processes (cf. Research Question 1) and, on the technological side, monitoring and incident response (cf. Research Question 2 and Research Question 3). Although ISO/SAE 21434 establishes a basis to meet the procedural requirements, and plenty of technological measures and tools are available, the discussion (cf. Chapter 5) has shown that several aspects are lacking to fulfill the defined objectives (cf. Chapter 3).

This dissertation proposes a cybersecurity operations approach for vehicles, which comprises several procedural (cf. ① and ② in Figure 6.1) and technological concepts (cf. ③ to ⑥ in Figure 6.1) to address the identified challenges. In line with industrial and scientific considerations (cf. Section 4.1.2), the approach centers around the introduction of a VSOC¹, responsible for the cybersecurity operations of all vehicles of an OEM. While this chapter provides an overview of the concepts and their relationships, the following chapters discuss the details. In particular, the cybersecurity operations activities in the VSOC and beyond are discussed in Chapter 7, and a new TTP-based method to define cybersecurity monitoring requirements in ISO/SAE 21434's concept phase is presented (cf. Chapter 8). Technological aspects include a holistic monitoring concept with information fusion in a Knowledge Graph (cf. Chapter 9), and immediate incident response over the air (cf. Chapter 10).

¹ cf. page 64

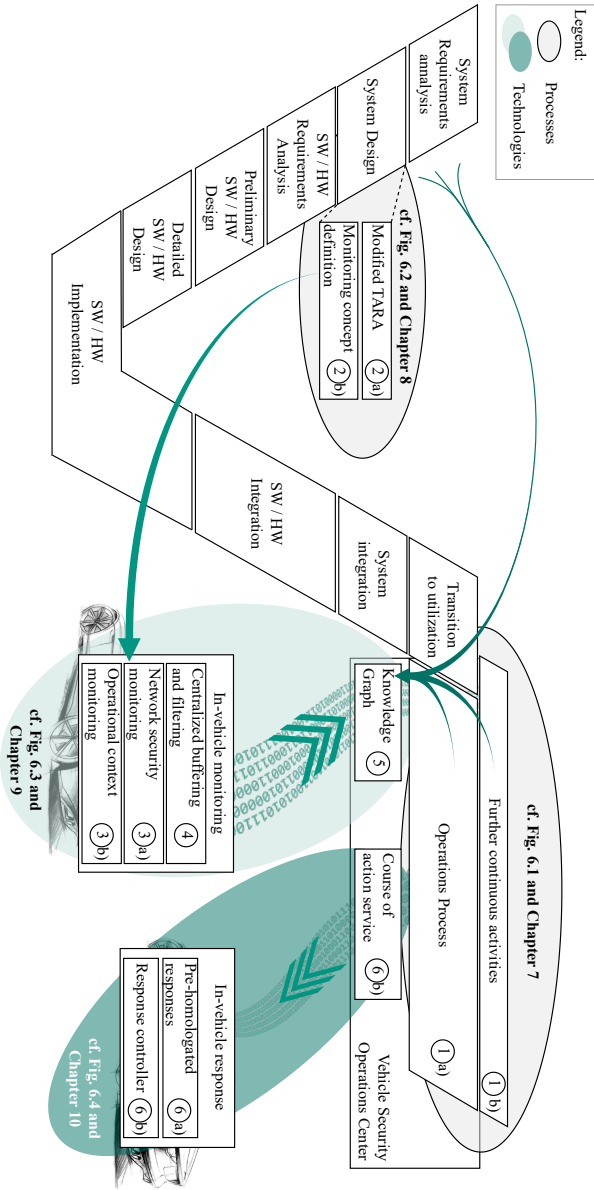


Figure 6.1: Overview of the proposed cybersecurity operations approach, refining the initial idea and requirements illustrated in Figure 3.1 on page 57. The referred Chapters detail the procedural (cf. Chapter 7 and 8) and technological (cf. Chapter 9 and 10) aspects of the proposed approach.

6.1 Processes

6.1.1 Process(es) for cybersecurity operations activities ①

Regarding idea one, two concepts are proposed that address the cybersecurity operations activities in the VSOC and beyond. The first concept is an operations process for the VSOC (cf. ①a) in Figure 6.2), second are complementary recurring activities outside the VSOC (cf. ①b) in Figure 6.2). The design of these procedural concepts is detailed in Chapter 7. Inspired by the DevOps model (cf. Figure 2.7 on page 25), a key aspect of the proposed processes is to connect the development process and cybersecurity operations (cf. Section 7.3.4), based on threat intelligence and TARA artifacts. In particular, the proposed procedure to define monitoring requirements (cf. Section 6.1.2) uses TTP-based threat information, but threat information can also serve to categorize incidents, and to share knowledge. Experience gained during operations can thus improve subsequent development processes, including continuous improvement of existing vehicles.

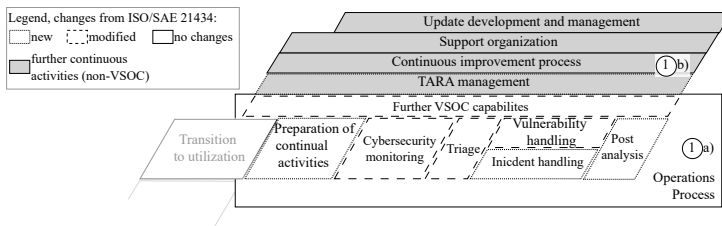


Figure 6.2: Overview of the procedural concept for cybersecurity operations, involving the operations process for the VSOC and activities beyond.

a) Operations process for the VSOC The comparison of ISO/SAE 21434 with the state-of-the-art in science and technology showed the need for procedural improvements for the operations phase (cf. Section 5.1.1). In particular, a new process is defined as the backbone for a VSOC (cf. ①a) in Figure 6.2), defining and connecting its tasks. Conceptually, the operations process follows the end

of the V-model and closes the identified gaps of ISO/SAE 21434 to established IT security processes. Besides the core VSOC capabilities (cf. Section 4.1.1), it includes new phases for preparation and post-analysis, aligns terminology and refines incident and vulnerability handling. Further VSOC capabilities, in particular threat intelligence management and information sharing, complete the operations process.

b) Continuous activities beyond the VSOC Besides, cybersecurity operations requires further continuous activities, that are not part of the VSOC (cf. ①b) in Figure 6.2). Some of these begin during development or are independent of particular development projects. Specifically, to ensure up-to-date risk assessments, a separate process manages the update of TARA artifacts, if threats were significantly misconceived. Update and configuration management is essential to fix vulnerabilities, and a continuous improvement process overarches the OEM's cybersecurity engineering activities. The established support structures (cf. support organization, Remark 4.1 on page 68) of workshops and help desks are also relevant for cybersecurity activities.

6.1.2 Procedure to specify monitoring requirements ②

The proposed procedure for specifying monitoring requirements, detailed in Chapter 8, is utilized during the system design phase of the V-model (cf. Figure 6.1). By specifying monitoring goals and a monitoring concept, the method extends the concept phase of ISO/SAE 21434, necessitating a modified TARA. Threat information plays a central role in the monitoring concept definition and the modified TARA.

a) Modified threat analysis and risk assessment Collecting cybersecurity information in the vehicle should contribute to the observability of identified risks. Hence, a suitable development method should yield risk-targeted monitoring requirements. Consequently, the proposed procedure modifies the TARA approach

(cf. ②a) in Figure 6.3) as defined in ISO/SAE 21434, which identifies the existing risks of an architecture. Threat information, particularly TTP, is used within the modified TARA to standardize the taxonomy of attack steps.

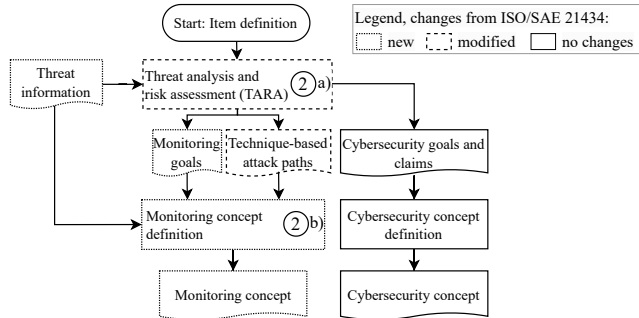


Figure 6.3: Overview of the proposed procedure to specify monitoring requirements.

b) Monitoring concept definition The monitoring concept definition step (cf. ②b) in Figure 6.3) specifies monitoring requirements and their allocation. First, the method maps attack steps to observable information by using threat information. Then, specifying and allocating requirements involves selecting suitable sensors that can collect the observable information. This results in the implementation of CMS and operational context monitoring, which are elements of the technological concepts presented in this dissertation (cf. ③ in Figure 6.1 and Section 6.2.1).

6.2 Technologies

The cybersecurity operations approach considers two technological issues: monitoring and incident response. An overview of the proposed concepts to address the ideas concerning monitoring (cf. ③ - ⑤) is provided in Section 6.2.1 to Section 6.2.3. The concepts are detailed in Chapter 9. By collecting and appropriately pre-processing the information from all identified views, the VSOC

achieves both technical and strategic CSA. This creates the necessary analysis context for decisions in the operations process, in particular to identify if incident response is required. If software updates to recover from the incident are not (yet) available, the last idea of this dissertation becomes momentous: to react immediately over-the-air (cf. ⑥). For this purpose, a solution concept based on pre-homologated responses is presented (cf. Section 6.2.4 and Figure 6.5), which is detailed in Chapter 10.

6.2.1 In-vehicle monitoring of cyber and physical aspects ③

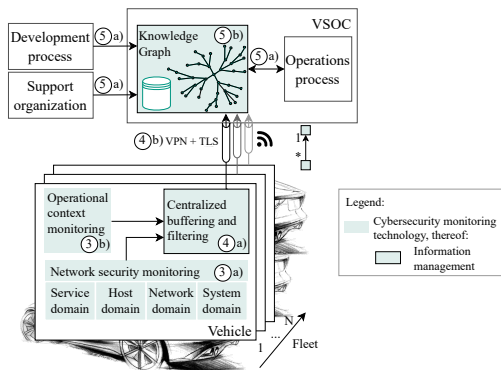


Figure 6.4: Overview of the monitoring concept, consisting of in-vehicle collection, secure transmission, and fusion with additional data in a Knowledge Graph. Information management has an n:1 relationship (vehicles:VSOC) due to the collection from the fleet.

a) Network security monitoring with system domain and SDV services First, the NSM terminology is extended with an additional monitoring domain describing the services required for the operability of the E/E architecture: the *system* domain. A taxonomy based on the the four NSM domains service, host, network, and system (cf. ③a) in Figure 6.4), defines common CMS. The information

generated by the CMS is mapped to a TTP model to support the monitoring requirements definition process (cf. Section 6.1.2). To utilize the service domain, a concept targeting containerized SOA applications is proposed. Since automotive applications are diverse, the concept integrates CMSs into the applications to collect logs.

b) Operational context monitoring Conceptually, operational context monitoring supplements NSM with a physical perspective (cf. ③b) in Figure 6.4). The definition of the operational context is derived from the ODC, which specifies safety boundaries. Since highly automated vehicles need to monitor the current operational conditions, the information could be utilized for further purposes. Operational context monitoring therefore collects for cybersecurity what already exists for safety.

6.2.2 Fleet-wide collection ④

a) Centralized buffering and filtering The third in-vehicle part is centralized buffering and filtering (cf. ④a) in Figure 6.4). All implemented CMS (cf. Chapter 8) and the operational context monitoring forward their information to a central entity. It converts the information to a standardized format for transmission to the VSOC (cf. Section 9.3.1). To use cellular bandwidth and the VSOC's handling capacity efficiently, only data that is currently deemed "required" is forwarded, configured by an adaptable *log level*.

b) Secure channel for monitoring data The entire fleet sends (pushes) cybersecurity information, including the operational context, via an over-the-air transmission channel to a central server or cloud of the VSOC operator. The operator of the VSOC can be the vehicle manufacturer or a service provider.

Securing the transmission is essential. Generally, a VPN² provides a secure connection between the vehicles and the manufacturer's network using encryption, and authentication (vehicle-level). However, compromised vehicles could infiltrate manipulated information into the fleet knowledge. If backend servers are compromised, the vehicles could send information to unauthorized recipients. Thus, in addition, the vehicle's centralized buffering and filtering component authenticates to the backend-side information management and vice versa using TLS (cf. ④b) in Figure 6.4).

6.2.3 Connection with additional information ⑤

While the vehicles NSM and operational context monitoring provide *technical* awareness, the *strategic* perspective is added by manufacturer-internal development data and the support organization, and the cybersecurity operations process. All monitored information is fed to a central database. The central data store provides cross-task (cf. Task T3 to Task T5) benefits, including its use as a single authoritative data source, viewing dependencies and interconnectivity, finding the responsible persons, and prioritization.

a) Development process, support organization, and operations process artifacts

Artifacts from the development process, including variant management, logical architecture, E/E architecture, and risk management, provide the development view and enable the instantiation of a vehicle shadow view per vehicle. The artifacts are the vehicles' documentation, particularly with regard to networks, critical assets, dependencies and responsibilities. Moreover, (automated) testing during development may identify weaknesses, referencing to CVE entries. The OEM's support organization bundles interaction with external sources that do not have an established relationship to share information, which includes the bug bounty program (cf. Remark 4.2 on page 84). The operations process adds artifacts to the database (old analyses, threat information, and weaknesses and

² Virtual private network (cf. glossary, page 330)

events from suppliers), but is also the user of the database by querying information required for analysis. Analysis results and any incident responses are documented in the database.

b) Semantic information fusion in Knowledge Graph The central database is part of the cybersecurity information management in the VSOC. Data collection and semantic fusion essentially serve to provide CSA for the VSOC. Key to this is a suitable data model or schema for normalizing the fleet's data and linking it with additional knowledge. The newly proposed model closes the gaps between the available models from different domains (cf. Table 5.3). Formalized as a KG (cf. ⑤b) in Figure 6.4), semantically rich query possibilities and flexibility are balanced.

6.2.4 Immediate response over the air ⑥

The primary response to incidents is currently the OTA update. With the development toward SDV, this thesis assumes an OTA update capability for all hosts in the vehicle (cf. Remark 2.3). For vulnerabilities that are not yet exploited (i.e. incident management is not required), but known early, *developing* an update may be feasible in terms of time. If a supplier already provides a patch, it can be tested and rolled out. As an update normally restores secure, full functionality, it implements the incident response strategy *Recovery* (cf. right side of Figure 6.5). Hence, the software update or configuration change remains the intended, final reaction to vulnerabilities and incidents.

a) Pre-homologated responses However, beyond OTA updates, this work proposes the homologation of *Containment*, *Eradication* and *Observation* responses during development (cf. ⑥a) in Figure 6.5). Such pre-homologated incident response actions enable the VSOC to respond more quickly, achieving an instantaneous risk reduction or increased observability. Besides the common approaches containment and eradication, the newly introduced *observation* responses shall

increase the attack visibility or check the effectiveness of a response. Moreover, this implements an element of the incident response plan (cf. *d*) in Remark 2.5). The response actions suitable for pre-homologation are analyzed, and respective actuators to implement the actions are discussed. In particular, the considerable actuators include security measures, the centralized buffering and filtering mechanism of the vehicle, execution management, and services.

b) Response activation over-the-air In contrast to autonomous approaches, such as fail-operational mechanisms for functional safety and IDPS for cybersecurity, incident response is managed from the backend (cf. ⑥b), response management component in Figure 6.5). However, while in enterprises, administrators provide remote support or employees are urged not to click on a phishing link, such interactions are not feasible in the vehicle. Instead, the proposed concept is to transmit reactions from the VSOC over-the-air to the vehicles (cf. ⑥b), response controller component in Figure 6.5) via a standardized interface. The response controller component triggers the respective actions of the actuators. The standardized interface of the vehicles' response control toward the VSOC

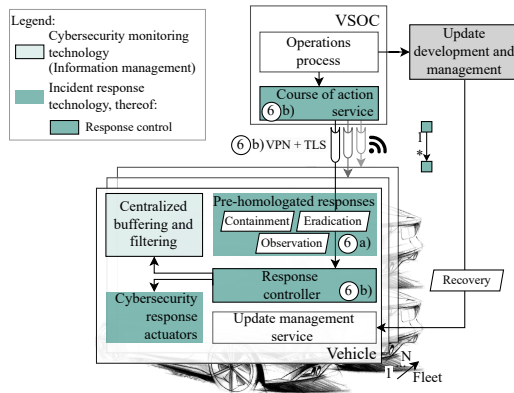


Figure 6.5: Overview of the response concept using pre-homologated containment, eradication and observation actions that target adaptation interfaces of cybersecurity response actuators and in-vehicle information management. Response control has an 1:n relationship (VSOC:vehicles) due to parallelized reaction in the fleet.

facilitates parallel incident response across the fleet, including different vehicle types.

As responses are security-critical changes, special attention must be paid to securing the interface toward the VSOC. Accordingly, the desired reactions are transmitted from the VSOC to the vehicles via the VPN plus TLS-based channel already used to collect monitoring data. In the vehicle, the actions are transmitted encrypted and authenticated from response control to the actuators.

7 Improving ISO/SAE 21434 for cybersecurity operations¹

Several modifications of the continual cybersecurity activities in ISO/SAE 21434 are proposed [Gri+23b]. To align the terminology with similar standards in IT security, the terms *cybersecurity event*, *cybersecurity incident* are updated, and the term *cybersecurity indicator* is introduced (cf. Section 7.1). Furthermore, the need for separation of vulnerability handling and incident handling is addressed (cf. Figure 7.1). These two activities are the core capabilities of the VSOC, which is why they are the focus of the cybersecurity operations process. Further continuous tasks complement the VSOC's core capabilities, in particular, threat intelligence management (cf. Definition 4.4), information sharing, employee training, and reporting. Threat intelligence management is particularly relevant to this work, as it is used in cybersecurity monitoring and definition of requirements. Moreover, aligning operations and development, this work proposes a new supporting *TARA management process* (cf. Section 7.3.1). This process has an impact on operations, but also on development. Updating TARA artifacts is a recurring task during operation, which makes it part of cybersecurity operations. However, as it is not a continuous, round-the-clock activity, this process is located outside of VSOC (cf. gray in Figure 7.2).

¹ This chapter is a rework of a previous publication by this dissertation's author [Gri+23b]: D. Grimm, A. Lautenbach, M. Almgren, T. Olovsson and E. Sax, "Gap Analysis of ISO/SAE 21434 – Improving the Automotive Cybersecurity Engineering Life Cycle," 2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC), Bilbao, Spain, 2023, pp. 1904-1911, doi: 10.1109/ITSC57777.2023.10422100. ©2023 IEEE.

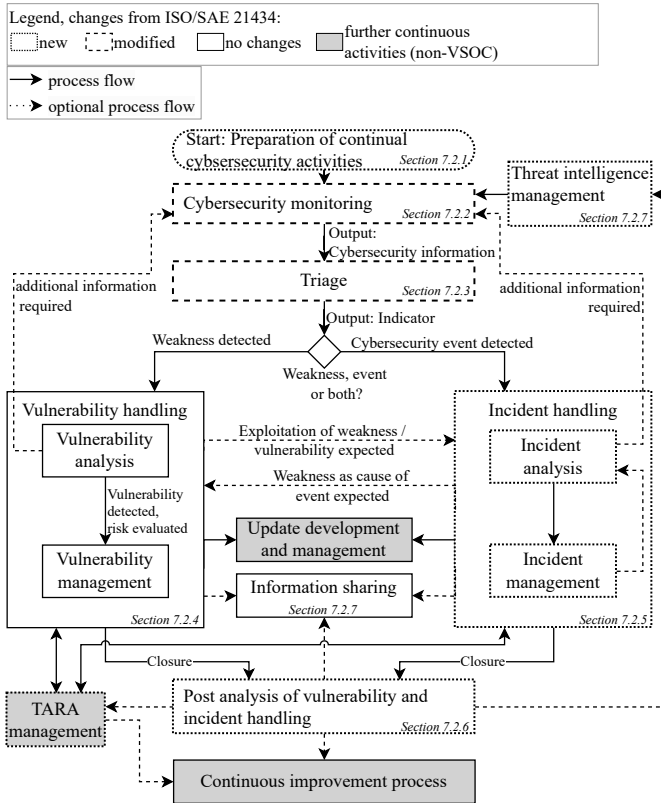


Figure 7.1: Proposed cybersecurity operations process.

7.1 Terminology: events, incidents and indicators

Since the terms *vulnerability* and *weakness* are in close alignment with similar standards, they are retained unchanged. While uncommon to be defined, the term *cybersecurity information* is retained as well (cf. Definition 2.26). However, the terms *cybersecurity event*, *cybersecurity incident*, and *cybersecurity indicator*

should be redefined. The considered definitions from six different sources are contained in the appendix (cf. Table A.1, A.2, A.3).

While in ISO/SAE 21434 a *cybersecurity event* is generically CSI that is relevant to an item or component, IT security-related documents define events in terms of *an occurrence in a system*. "Relevant cybersecurity information" could be a recently discovered vulnerability in software, but that would not fulfill the definition of an event according to IT security terminology. Accordingly, an adaptation of [Shi00] and ISO/SAE 21434 is proposed:

Definition 7.1: (Redefinition) Cybersecurity event

An occurrence in an item / component that is relevant to the cybersecurity of the system.

This redefinition ties an event to something happening in a system, rather than being any kind of CSI. This makes an event a precursor to an incident, just like a weakness is a precursor to a vulnerability.

ISO/SAE 21434 defines a *cybersecurity incident* as a "*situation in the field that can involve vulnerability exploitation*", which is both too broad and too narrow at the same time, depending on the interpretation of "can". In IT security, incidents have varied definitions, but a common theme is the concrete breach of a security policy, and it is often tied to security events. The following redefinition is proposed:

Definition 7.2: (Redefinition) Cybersecurity incident

A cybersecurity event that involves a violation of product cybersecurity, product safety or organizational cybersecurity policy.

By tying the incident to an event, it is tied to "an occurrence in a system", making it more concrete than the previous definition. By broadening the scope to organizational cybersecurity policy, it also accounts for breaches during development.

With the redefinition of cybersecurity event, there is no longer an overarching term for relevant CSI. The usefulness of a term for this purpose is not at issue,

since it can serve as an umbrella term for both events and weaknesses. Here, the term *cybersecurity indicator* is proposed for the same definition:

Definition 7.3: Cybersecurity indicator

Information with regard to cybersecurity that is relevant for an item or component.

7.2 Operations process for a VSOC

To address the identified gaps in ISO/SAE 21434, a comprehensive update to the continual activities is proposed. The new process (cf. Figure 7.1) introduces preparation and post-analysis activities, and splits vulnerability and incident handling into separate processes. In addition, the sub-processes "cybersecurity monitoring" and "event evaluation" are renamed and clarified.

7.2.1 Preparation

ISO/SAE 21434 indirectly requires preparation as part of [RQ-05-02] to [RQ-05-05]. However, it is beneficial to explicitly account for the preparation of processes, rules, capabilities and responsibilities for vulnerability and incident handling, because in the broader context of cybersecurity governance where it otherwise fits in ISO/SAE 21434, it may be overlooked. Several work products should be prepared for the continual cybersecurity activities (cf. Table 7.1).

Based on NIST SP 800-61 [NIS12], for vulnerability & incident analysis, it is recommended to prepare communication channels to in-house responsible persons for the different products (e.g., components, vehicle model), required hardware and software for analysis tasks, and access to vehicle configuration data. The resources should cover documentation of the vehicle's network architecture and protocols, operating systems, software, hardware, and its cybersecurity concept. The resources could be supplied as documents or tables. However, this work

Table 7.1: Work products of the Preparation phase.

Target phase	Work product
General	Digital evidence retention policy and facilities, exchange formats and information sharing policy (with TARA management and externally)
Triage	Triggers for validation, methods for categorization and prioritization
Vulnerability & incident analysis	contact data of internal experts and responsible persons, access to configuration management (build and development environment, SW/HW versions), access to documentation of assets (network diagrams, functions, proprietary protocols, security mechanisms, monitoring)
Incident management	Incident response plan template, response strategies, escalation criteria

pursues a comprehensive approach by providing all resources mentioned via a KG (cf. Section 9.4.3). In addition, IT security typically compares current network behavior against a historical network baseline. In the automotive sector, this would require a baseline per-vehicle model or per-variant.

For incident management, an organization-level template for the incident response plan (cf. Section 5.1.1) should be prepared. It is expected that elements *a)* to *c)* of [RQ-13-01] in ISO/SAE 21434 (cf. Section 2.3.2) require per-incident planning, whereas *d)* to *g)* can be prepared on an organizational level. Accordingly, the template should define: a method for collecting additional information on an incident, a method for determining progress, and criteria and actions for closing an incident. A *technology* to implement methods for collection of additional information is proposed in this work (cf. Section 9.3.1 on page 184).

7.2.2 Cybersecurity monitoring

In ISO/SAE 21434, *cybersecurity monitoring* means "collecting cybersecurity information and analyzing it for triage based on defined triggers". A slight scope change is proposed by separating triage into its own activity with the following re-definition of the term "monitoring":

Definition 7.4: (Redefinition) Cybersecurity monitoring

The process of collecting, pre-processing and formatting cybersecurity information for subsequent processing.^a

^a This definition has already been presented in Chapter 1 and is mentioned again here for the structure of the chapter.

Pre-processing and formatting result in a uniform data format to facilitate human understanding and automatic processing. Depending on the source, different steps may be required to prepare the information for subsequent processing. For example, a vulnerability report may be parsed manually, whereas production data may be queried from a database. Monitoring also considers information not inherently cybersecurity-related, but to be linked to cybersecurity information for further processing (e.g., variants, versions), thus becoming second-degree CSI.

ISO/SAE 21434 lists internal and external data sources (cf. Figure 2.15 on page 51) to consider, addressed in detail by the proposed monitoring concept (cf. Section 9.1). Process-wise, the confidentiality of information (i.e. public or private) should be considered, since it affects the urgency to address it.

The process (cf. Figure 7.1) has a feedback loop from vulnerability and incident handling back to cybersecurity monitoring, since new technical details may become available during the handling processes, such as details about the vulnerability or incident, or about additional sources to be monitored.

7.2.3 Triage

Triage is defined as "Analysis to determine the relevance of cybersecurity information to an item or component" in ISO/SAE 21434, which clashes with typical definitions (cf. Section 5.1.1) as it does not include prioritization. The following redefinition is proposed:

Definition 7.5: (Redefinition) Triage

The process of determining relevance of cybersecurity information and prioritization of resulting indicators to facilitate their appropriate handling.^a

^a This definition has already been presented in Chapter 1 and is mentioned again here for the structure of the chapter.

In essence, *triage* replaces *cybersecurity event evaluation*. The input for triage is the pre-processed CSI, and it has three components: (1) determining relevance, (2) categorization and (3) prioritization. Determining relevance may include searching for or confirming affected assets, assessing confidence in the information source, or verifying the information. If the information is not relevant, i.e. not an *indicator* (cf. sec. 7.1), it should still be archived for post-analysis later. Once an indicator has been identified, it should be categorized as being a weakness, a cybersecurity event, or both. Triage usually involves correlation [Alb+04], which groups new indicators with already categorized vulnerabilities or events. Prioritization may be based on metrics such as the number of affected vehicles or potential impact, e.g. a CVSS score. Because detailed information may still be missing, at least a preliminary prioritization is expected. Categorization and prioritization may require updates during subsequent activities.

7.2.4 Vulnerability Handling

If the outcome of *triage* is a weakness, *vulnerability handling* is triggered. Minor adjustments to ISO/SAE 21434 are applied in this aspect. Vulnerability handling comprises vulnerability analysis and vulnerability management (cf. Figure 7.1):

Definition 7.6: Vulnerability handling

The process of performing vulnerability analysis and vulnerability management.

Definition 7.7: Vulnerability analysis

The process of identifying and assessing vulnerabilities from identified weaknesses, which includes performing the appropriate steps of threat analysis and risk assessment.

Definition 7.8: Vulnerability management

The process of tracking and treating cybersecurity vulnerabilities.

Vulnerability analysis investigates weaknesses to identify root causes and risks associated with exploitation, which requires interaction with the proposed TARA management process (cf. Section 7.3.1 on page 143). While the TARA is refined during development (cf. TARA levels outlined in Section *ISO/SAE 21434 in practice* on page 69), the reverse can be expected during vulnerability analysis: TARA updates require abstraction, from fine-grained to coarse. Vulnerabilities often boil down to a line of code that leads to a potential attack path. From such an attack path at the sub-component level (detailed), tracking the TARA levels in the TARA management process can infer effects (threats) at the upper levels (e.g. system level). If the analysis detects a vulnerability for which no update is available, the attack steps should be included in cybersecurity monitoring, to observe potential exploitations. This may involve a software or configuration update, if the appropriate CMS are not in place, and observation actions to raise forwarding (cf. Section 9.3.1 on page 184) of the current CMS are not sufficient. If the indicators show active exploitation of a vulnerability, vulnerability handling triggers incident handling, since an immediate response may be necessary. Depending on the policy (cf. Table 7.1), informing private stakeholders and the public about found vulnerabilities may be required, which triggers information sharing (cf. [RQ-05-09] and [RQ-05-10] in ISO/SAE 21434).

Vulnerability management includes vulnerability treatment and tracking its resolution. Implementing a solution, however, is done in the related process *update development and management*. Solutions could include the development of a software or configuration update for existing components, or integration of an

available patch. The updated associated risks should determine the urgency of solution deployment.

7.2.5 Incident Handling

Incident handling is introduced as a separate process. It is triggered if the outcome of *triage* is an event. The incident response provisions in ISO/SAE 21434 are minimal. The extension proposed here loosely builds on NIST SP 800-61 [NIS12] and 2004-TR01 [Alb+04]. Structurally similar to vulnerability handling, it comprises incident analysis and management:

Definition 7.9: Incident handling

The process of analysing and managing cybersecurity incidents.

Definition 7.10: Incident analysis

The process of identifying and assessing cybersecurity incidents from identified cybersecurity events.

Definition 7.11: Incident management

The process of tracking and treating cybersecurity incidents, including actions for containment, eradication or recovery.

The goal of incident analysis is not to find the root cause, but to understand an event in sufficient detail to determine if it indeed is an incident, and if it is, to collect enough information to be able to prepare a response. This may require additional data collection, for which there is a loop to cybersecurity monitoring. Additional in-vehicle data can be collected with observation actions (cf. Section 9.3.1).

Once an incident has been identified, incident management determines an action plan and tracks its progress. The actions should include a plan for containment, eradication, and recovery, or a subset that leads to full recovery. Actions may include temporary function degradation. It may also be necessary to come up

with alternative, ad-hoc measures, especially if the initial remedial actions turn out to be unsuccessful. To that end, this work proposes pre-homologated actions (cf. Chapter 10) to minimize the amount of ad-hoc measures affecting type approval. Incident handling is an iterative process and additional analysis may become necessary at any point. Accordingly, a loop exists from incident management back to analysis (cf. Figure 7.1). As with vulnerabilities, the final remedial actions (recovery) involve *update and configuration management*.

Incident handling results in an implemented incident response plan, based on the template (cf. Section 7.2.1). Actions, responsibilities, and communication strategy need to be planned and implemented per incident. As with vulnerability handling, information sharing may be necessary. If an incident is caused by a new vulnerability, vulnerability handling is triggered. In that case, incident and vulnerability handling can run in parallel, although a steady exchange of information is expected.

7.2.6 Closure and post-analysis of vulnerability and incident handling

Vulnerability or incident handling is closed when no further action is required, in particular when:

- The incident or vulnerability has been resolved with an update that was successfully rolled out and verified.
- The incident or vulnerability has not been fully resolved, but the accompanying risk is sufficiently low to decide that no further actions are required.
- The event or weakness has been judged as not being an incident or a vulnerability, respectively.

Closure triggers archiving and final notification of stakeholders. What needs to be archived is case-specific. Even if triage concludes that no vulnerability or

incident handling is required, i.e., no indicator is present, it may be useful to keep the information temporarily.

After closure, a meta-analysis of the processes and results should be performed. One could argue that the continuous improvement process ([RQ-05-08]) already covers that, but this work argues that a separate activity is needed to enable effective continuous improvement.

Definition 7.12: Post-analysis of vulnerability and incident handling

The process of analyzing and measuring incident and vulnerability handling outcomes, including root cause analysis and efficiency measurements.

A root cause analysis is recommended in post-analysis. For incidents, this is necessary if it was not performed during incident handling. The technical root cause of vulnerabilities should have been investigated already, but how it was introduced may not be known. This could reveal weaknesses in development or testing processes. If new problems are identified in the course of the root cause analysis, repeated (external) information sharing may be necessary.

The second recommendation is measuring process efficiency, process validation and threat intelligence. Appropriate organization-specific measures need to be defined, for example the duration to measure efficiency, measuring the number of identified vulnerabilities for validation, or categorizing the types of incidents for threat intelligence. Post-analysis may also reveal improvement potential to cybersecurity monitoring. Note, that potentials may also be identified via TARA updates (cf. Section *TARA management as a continual activity* on page 145).

Drawing conclusions to improve the processes, the vehicle's security architecture, and monitoring takes place in the subsequent continuous improvement process (cf. Section 7.3.4). Reviewed incidents and metrics are also an input for the CTI capability (cf. Figure 9.1 on page 166), which may update the threat information used within the operations process.

7.2.7 Further VSOC capabilities

Among the further capabilities of the VSOC, information sharing and threat intelligence management are involved in the proposed process (cf. Figure 7.1). Threat intelligence management (cf. Definition 4.4) is one of the essential sources of information for monitoring, as it provides strategic perspective to classify technical information from the field (cf. Chapter 9) and can be useful for configuring SIEM rules [Olt18]. External sources such as the deep web (especially forums where vulnerabilities and exploits are discussed), public sources (such as social media and press reports) and the dark web are used for this purpose. In addition, the company's own post-analyses are a source for improving threat intelligence.

ISO/SAE 21434 mandates information sharing, and the corresponding requirements [RQ-05-09] and [RQ-05-10] are not changed. Essentially, the task is therefore to inform relevant (external) stakeholders about identified vulnerabilities and incidents. In addition, this thesis proposes to include the exchange of information with suppliers under this task, which is not considered in ISO/SAE 21434. While the cybersecurity interface agreement contractually defines the requirements and content of the information exchange, the coordination of the exchange should be part of an information sharing capability of the VSOC. Due to the large amount of different hardware, software and suppliers, where, in contrast to IT systems, vulnerabilities often cannot be obtained from public databases, it makes sense to define a dedicated responsibility. A further task is the exchange with closed communities such as ISACs. The information obtained from information sharing either flows into monitoring as reports or is analyzed in threat intelligence management (not shown in Figure 7.1, can be seen in Figure 7.3).

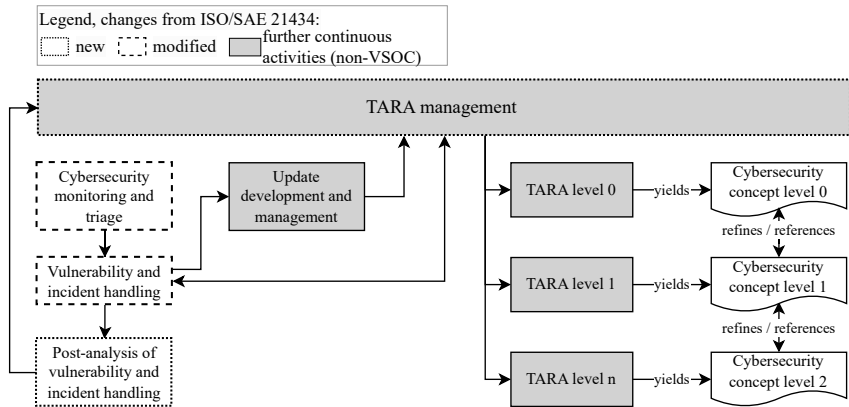


Figure 7.2: Interfaces to the TARA management process.

7.3 Further continuous activities beyond the VSOC

7.3.1 TARA management

Conceptually, the TARA management process (cf. Definition 7.13) is a continual cybersecurity activity because it spans the entire lifecycle of a project and defines an organizational interface for exchanging TARA information across projects. It addresses keeping risk assessments up to date, including distributed and multi-level development.

Definition 7.13: TARA management process

The process of brokering access to and coordinating work on TARA artifacts required for an item, including the coordination of distributed development activities and the definition of appropriate abstraction levels.

Brokering access to artifacts

The management process governs access to TARA artifacts. Specifically, it outlines who has access to which parts of the TARA under which circumstances. As such, it acts as an *artifact access broker* to other processes, such as vulnerability and incident handling (cf. Figure 7.2). Most TARA artifacts are project specific, but the TARA management process has an organization-wide scope, allowing for information exchange and artifact reuse between projects. Another challenge that the management process addresses is the interaction with suppliers, as it coordinates which information is shared with suppliers and when.

The refinement of TARA over several levels is elementary for the development, as suggested in several publications [Dob+23; Lie+24] (cf. Section *ISO/SAE 21434 in practice* on page 69). Dieser aus der Literatur bekannten Ansatz wird in dieser Dissertation ebenfalls angenommen. During the concept phase (cf. Clause 9, Figure 2.10 on page 42), the technical realization is not yet available, so TARA can only lead to high-level cybersecurity goals. As soon as the development process proceeds (cf. Product Development phase, Clause 10 in Figure 2.10) to component or sub-component design (software, hardware), a more detailed TARA can be performed. However, the level of detail required can no longer be represented in a single diagram. A single DFD, which is often used for TARA, would be insufficient to adequately describe a vehicle. Thus, a multitude of diagrams, per component or sub-component, and possibly additional diagrams for modes or variants of the components, must be generated. This requires a management process that governs tracking of the artifacts. The TARA management process interfaces with the TARA artifacts (cf. Figure 7.2), including the refinement steps. The cybersecurity concept on level $n + 1$ is a refinement of the cybersecurity concept on level n , tracked by the TARA management process.

TARA management as a continual activity

Over the vehicle's lifecycle, the risks cannot be considered static, so the TARA management process must be continual (cf. Figure 6.1). Several relationships between the continual activities from ISO/SAE 21434 with the TARA management process exist (cf. Figure 7.2). They depict a subset of the proposed cybersecurity operations activities (cf. Section 7.2). TARA updates (cf. arrows toward the management process in Figure 7.2), are required in several cases, e.g. when a software or hardware update is developed. After updating TARA artifacts, a re-evaluation of the requirements in the continuous improvement process (cf. Figure 7.1) is necessary in accordance with the initial development process, including the cybersecurity monitoring requirements. TARA updates may be required based on vulnerability handling findings, since new weaknesses or vulnerabilities introduce risks. There are two cases to consider:

1. There is at least one attack path known in which the weakness may be exploited, or
2. there is no known attack path that includes the weakness.

In the first case, the attack feasibility rating (cf. Definition 2.22 on page 44) and risk value (cf. Remark 2.4 on page 43) are updated if needed. The second case indicates that a threat was overlooked.

Incident handling can provide new insights regarding assumptions of exploitability and impact. Mapping events to TARA artifacts is not trivial, since the type of monitored CSI determines if it affects an attack path or a damage scenario. Consequently, if the event affects an attack path, the same options apply as for updating TARA artifacts based on new weaknesses. Similarly, two cases can be distinguished for damage scenarios:

1. There is at least one known damage scenario in which the event may be an observable indicator, or
2. there is no known damage scenario related to the observed event.

In the first case, the impact rating (cf. Definition 2.25 on page 46) is updated. In the second case, a new damage scenario and potentially a new threat scenario with associated risk value (cf. Remark 2.4) must be added. To simplify mapping, this work proposes to model attack paths using TTP (cf. Chapter 8), and to connect development (TARA) and operations (incident handling) with corresponding CMS (cf. Figure 7.3). In the case of an unknown attack path, updating the TTP model may be required if it is not suitable to represent the attack step (cf. Section 7.3.4).

Finally, the new step of post-analysis may inform about frequent sources for weaknesses or recent or typical attack vectors. Since TARA ratings (attack feasibility, impact) are based on assumptions about frequency and difficulty, post-analysis may yield updates of TARA assumptions based on real-world observations.

7.3.2 Support organization

The proposed support organization is based on the approach of Harde et al. [HG18] (cf. Figure 4.4) in major parts. For end customers and the public, such as the press or government organizations, the infrastructure is already well-established. Customers can contact workshops or a hotline of the OEM, and there are specific contact points for public relations and legal matters. The staff serving these interfaces should therefore be trained to identify potential cybersecurity-related information and, if necessary, forward it to the VSOC. The received information is usually unstructured, such as reports, which must be pre-processed accordingly (cf. Section 7.2.2). A special point of contact for responsible disclosure including a bug bounty program should be set up, which is also part of the support organization. In contrast to Harde et al., however, suppliers should not contact an anonymous support organization, as this is already coordinated by the information sharing capability of the VSOC (cf. Section 7.2.7).

7.3.3 Update development

In IT cybersecurity, the specification and maintenance of monitoring and prevention capabilities can be a task of the SOC [KPZ23]. In contrast, as vehicles undergo a development process, the specification is a task of the development team in the automotive domain. Although other works consider asset and change management a VSOC's capability [HBH23], this thesis argues that maintenance activities, including update development and change management, are original tasks of the development team. In particular, [RQ-05-11] and [RQ-06-12] of ISO/SAE 21434 mandate management of configuration, change, requirements, and documentation already for the development phase. Keeping these tasks at variant and requirement management experts and tools avoids overloading the VSOC's competencies.

7.3.4 Continuous improvement

The cybersecurity operations process is interlinked with development in several ways (cf. Figure 7.3), enabling long-term maintenance and improvement of preventive and monitoring capabilities. First, in the concept phase of a new vehicle development, the TARA is carried out and initial cybersecurity requirements and monitoring requirements are defined (cf. Chapter 8). The VSOC advises this phase and provides threat information (TTP), which describes attacker behavior. Updating the threat information is the task of the VSOC within the threat intelligence management capability. While OEMs have to start with an initial knowledge of threat information, e.g. derived from public sources such as the ATM [SC24], CTI improves from post-analysis of incidents over time. Together with knowledge from external sources such as information sharing, updated threat information flows into new development projects and cybersecurity monitoring.

In addition, TARA may be updated via the TARA management process after post-analysis or triggered from incident and vulnerability handling. This improves the risk assessment continuously and keeps it up-to-date, but can also lead to reconsidering requirements. Identified improvement potentials are incorporated

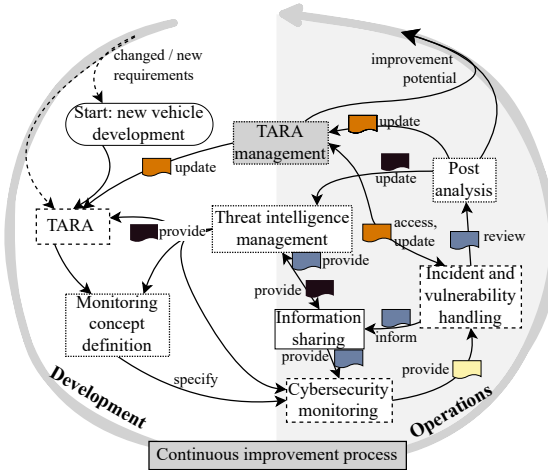


Figure 7.3: DevOps-inspired workflow for the continuous improvement process and its integration with TARA and cybersecurity operations activities.

into the continuous improvement process by the TARA management process. In parallel, the post-analysis phase also suggests improvements identified from incident and vulnerability handling. New or changed cybersecurity requirements and monitoring requirements that are defined in the continuous improvement process are then implemented by the development team.

By differentiating between the analysis of improvement potential (post-analysis and TARA management) and improvement decisions (continuous improvement process), the responsibilities within the company are clarified. The post-analysis is the task of the VSOC, while the improvement potentials are implemented in product development or corporate IT, driven by prioritization within the continuous improvement process. This involves the VSOC's knowledge in further development of cybersecurity operations (cf. Section 7.2.6), while limiting the scope of its tasks.

8 A threat information-based procedure to specify cybersecurity monitoring requirements

Monitoring the in-vehicle network is a core cybersecurity operations task of an OEM. Accordingly, the specification of monitoring requirements should have the same importance as (preventive) cybersecurity requirements. To supplement the risk-based derivation of cybersecurity requirements within the TARA, a development procedure to specify monitoring requirements should focus on observing risks instead of collecting every possible information. From the literature (cf. Section 4.1.4 on page 73), the ACF¹ is the most promising approach to leverage, as its first two steps, identifying threats and quantifying associated risks, correspond to TARA (cf. Figure 8.1). Thus, compared with ISO/SAE 21434, a procedure to specify cybersecurity monitoring requirements inspired by the ACF would require two further steps - *Identify Data Feeds* and *Narrow focus*.

The following section integrates *identify data feeds* in ISO/SAE 21434's concept phase, resulting in the *monitoring concept* as a new work product. In parallel to the cybersecurity concept (cf. [WP-09-06] in ISO/SAE 21434), the monitoring concept specifies a set of suitable sensors and, if possible, their allocation to components. Similarly to cybersecurity goals and requirements in ISO/SAE 21434, *monitoring goals* and *monitoring requirements* are differentiated. A modified TARA yields monitoring goals as high-level monitoring requirements. The monitoring concept definition step (cf. Section 8.2) refines the goals to technical item-level monitoring requirements. *Narrow focus* refines the item-level

¹ cf. page 73

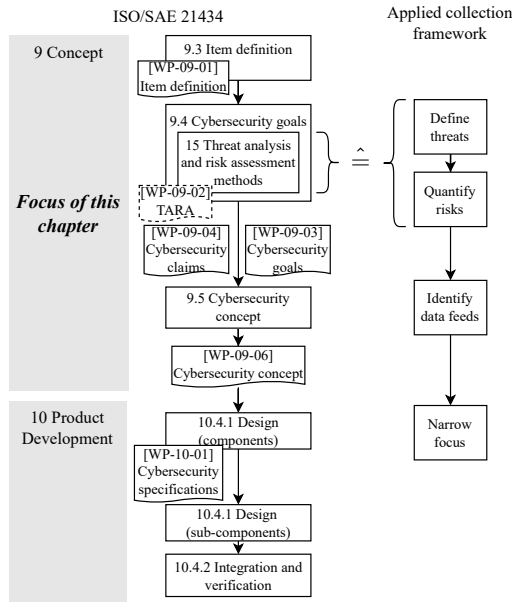


Figure 8.1: Mapping ISO/SAE 21434’s development phase (left, cf. Figure 2.10) to the ACF (right, cf. Figure 4.6).

requirements to component-level requirements by selecting specific information to monitor. This step is part of the product development phase of ISO/SAE 21434 (cf. Figure 8.1). For example, if the monitoring concept specifies a NIDS, the product development phase specifies if it is rule-based or anomaly-based and the algorithms for detecting given threats. This chapter focuses on *identify data feeds*, leaving the *narrow focus* phase to future work. This chapter proposes several additions to ISO/SAE 21434’s concept phase to incorporate the identification of data feeds (cf. Figure 8.2): A catalog of threat information mapped to observable information (*I*), a modified TARA (*II*), yielding monitoring goals (*III*) and technique-based attack paths (*IV*), and a monitoring concept definition step (*V*), yielding the monitoring concept (*VI*). The appendix summarizes the nomenclature of terms used in this chapter (cf. page 331).

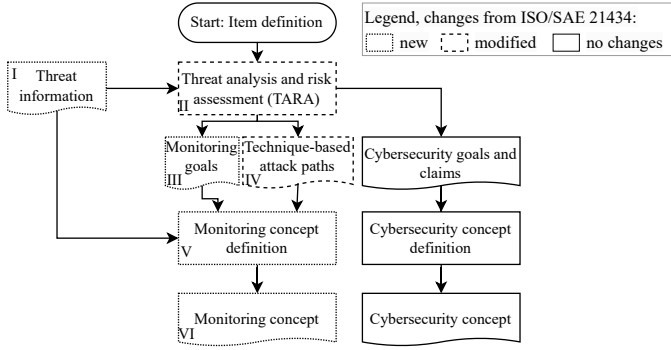


Figure 8.2: Overview of the extended ISO/SAE 21434 concept phase (cf. Figure 2.10) to identify data feeds.

8.1 TARA with threat information

The starting position for the modified TARA (cf. *II* in Figure 8.3) proposed in the following is ISO/SAE 21434’s approach. To identify the threat scenarios, STRIDE² is used on a DFD model of the item-related network, as suggested in ISO/SAE 21434 and literature [LAO21; LI16] (cf. Section *Example: HEAVENS 2.0 workflow* on page 47). DFD models fit well with the logical architecture (L in RFLP, cf. Section 2.2.2), corresponding to the concept phase (cf. Figure 2.9). The central addition to the original workflow is a TTP-based catalog of techniques and tactics (cf. *I-a*) in Figure 8.3), that yields a uniform taxonomy for TARA’s analysis steps. θ_i is a specific tactic in the I tactics in Θ , where

$$\Theta = \{\theta_i \mid i \in [1, I]\}, \quad (8.1)$$

and t_j is a specific technique of the J techniques in $\mathcal{T}$:

$$\mathcal{T} = \{t_j \mid j \in [1, J]\}. \quad (8.2)$$

² cf. 44 and Table 2.1

Then, a TTP catalog or matrix $\mathcal{C}$ is defined as

$$\mathcal{C} := \{(\theta_i, t_j) | \theta_i \in \Theta, t_j \in \mathcal{T}\}. \quad (8.3)$$

$\mathcal{C} \subseteq \Theta \times \mathcal{T}$ is a subset of the cartesian product of Θ and $\mathcal{T}$, i.e., only particular tuples (θ_i, t_j) associate techniques with particular tactics. $(\theta_i, t_j) \in \mathcal{C}$ implies the technique t_j is *used in* the tactic θ_i . Procedures (the "P" in TTP) are not relevant for the identification of monitoring requirements.

Several automotive-focused TTP catalogs have been published (cf. Section *Public and open sources* on page 82) but are not yet proven in practice and not integrated with ISO/SAE 21434. A tool developed in this dissertation (cf. Section A.2) implements an initial catalog and related observables derived from the literature, whose details can be found in the appendix (cf. Table A.4, Table A.5 and Table A.6).

In the following, the modified TARA models the threat and damage scenarios (cf. *II-a* and *b*) in Figure 8.3), and attack paths (cf. *II-c*) and *IV* in Figure 8.3) as attack steps (cf. Definition 2.23) of particular techniques t_j of the TTP catalog $\mathcal{C}$. The result of classifying the TARA artifacts as techniques t_j is a unified naming and meaning. Based on a modified risk treatment decision, which offers the option to increase observability (cf. *II-d* and *e*) in Figure 8.3), monitoring goals are subsequently defined (cf. *III* in Figure 8.3).

8.1.1 Threat scenario and damage scenario identification (*II-a*, *II-b*)

After identifying the threat scenarios $v_k \in \Upsilon, k \in [1, K]$ with STRIDE, they are classified as techniques from the catalog $\mathcal{C}$ (cf. *I-a*, Figure 8.3). For any threat scenario $v \in \Upsilon$, where Υ denotes all threat scenarios of the item under consideration, a classification T_v is defined as

$$T_v : \Upsilon \mapsto \mathcal{T}, \text{ where } T_v(v_k) = t_j, t_j \in \mathcal{T}. \quad (8.4)$$

For example, if v_k is *Denial of service of brake pedal position information by flooding the bus*, it could be classified as $t_j = \text{Bus Denial of Service}$. Optimally, the techniques and sub-techniques in the catalog (*I-a*) are annotated with STRIDE's threat classes and threatened assets (data in rest, data in motion, software, hardware) corresponding to the threat scenario identification. While STRIDE analysis per asset (DFD elements) identifies what can go wrong (e.g., spoofed communication, or tampered software), classifying threat scenarios as techniques improves the identification of how attackers can exploit these threats. The catalog $\mathcal{C}$ (cf. *I-a*) in Figure 8.3) requires an update if it does not contain a

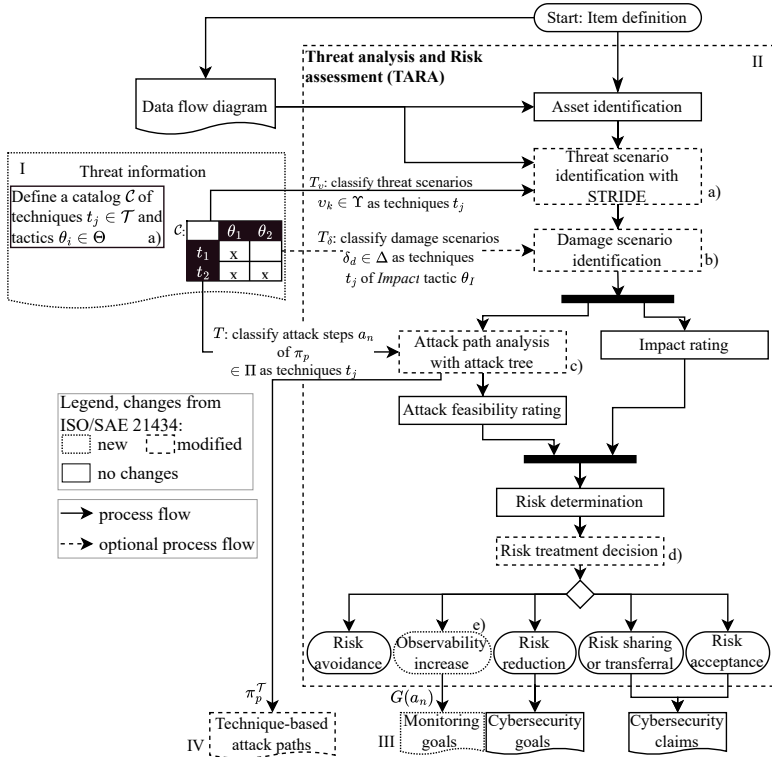


Figure 8.3: Modified TARA with threat information (refines Figure 2.12 on page 47 [LAO21]).

matching technique (decision if update of catalog is required is not modeled and out of scope).

Subsequently, the identification of damage scenarios (cf. *II-b*), Figure 8.3) can be guided with the *Impact* tactic θ_I^3 . The techniques used in the *Impact* tactic represent damage scenario categories, such as *Loss of Safety*. For any damage scenario $\delta_d \in \Delta, d \in [1, D]$, where Δ denotes all damage scenarios of the item under consideration, the classification T_δ of damage scenarios δ_d to techniques t_j can be defined:

$$T_\delta : \Delta \mapsto \mathcal{T}, \text{ where } T_\delta(\delta_d) = t_j \text{ such that } (\theta_I, t_j) \in \mathcal{C}. \quad (8.5)$$

For example, if the damage scenario δ_d is *Malfunction of brake due to not receiving braking signal*, the damage scenario could be classified as $t_j = \textit{Denial of control}$. Although optional for the identification of monitoring requirements (cf. dashed arrow in Figure 8.3), considering the scenario categories is beneficial, as it limits the scope of potentially overlooked damage scenarios.

8.1.2 Attack path modeling (*II-c, IV*)

The main aim of modifying TARA is the classification of attack paths using techniques. As suggested in the literature [ISO21; LAO21], attack paths are modeled using attack trees. In particular, model-based design benefits from attack trees because they can be broken down into reusable sub-models and represented graphically. Damage scenarios δ_d are chosen as the root nodes of the trees [LAO21]. Below the root nodes, the threat scenarios causing the damage scenarios follow. A sub-tree is created for each threat scenario v_k , adding attack steps a as further nodes to the tree. A denotes the (unordered) set of attack steps a_n of the item (all nodes of the attack tree), including the threat and damage scenarios v_k and δ_d , where $a_n \in A, n \in [1, N]$. The set of unique paths π_p from the leaf nodes to the damage scenario for all attack trees yields the total set of

³ The *Impact* tactic is usually the last (*I*-th) tactic of a catalog [Yek+23; Str+20]

attack paths Π . Hence, an attack path π_p is a finite sequence of N_p attack steps $a_n \in A, n \in [1, N], N_p \leq N$, where two paths $\pi_p, \pi_q \in \Pi, \pi_p \neq \pi_q$ can have certain attack steps $a_n \in A$ in common:

$$\pi_p = (a_1, \dots, a_{n-1}, a_n, a_{n+1}, \dots, a_{N_p}). \quad (8.6)$$

In an attack path π_p , the a_n are ordered as $a_1 \prec \dots \prec a_n \prec \dots \prec a_{N_p}$. To achieve the attack step a_n , the preceding step a_{n-1} must be achieved first. a_{N_p} is the damage scenario δ_d and a_{N_p-1} is the threat scenario v_k of a particular path. In sum, for any item, there may exist several threat scenarios v_k in Υ , damage scenario $\delta_d \in \Delta$, and attack paths $\pi_p \in \Pi, p \in [1, P]$.

Attack paths originating from remote interfaces (e.g., wireless) are typically high-risk and should be prioritized in attack path analysis [Lie21; LAO21]. Since attack trees represent the structured progression of an attack — from initial entry points at the deepest levels to impact at the root — the analysis should account for this hierarchical order. TTP catalogs follow a sequential structure, with tactics progressing from initial access to later-stage actions. For example, the techniques used in the tactics "initial access" and "manipulate environment" can initiate external attack paths. The ordering of tactics reflects logical attack sequences. For instance, *lateral movement* is typically preceded by *privilege escalation* to obtain the necessary write permissions. Constraining the currently considered techniques to tactics relevant to a given depth in the attack tree aligns the attack path analysis with this progression. The TTP-aligned attack path analysis enables a systematic identification of critical attack paths.

Since there is no common approach to formulate the steps in attack trees, this thesis suggests describing each step in a granularity matching the techniques, enabling a classification into a technique. For example, the attack step a_1 *Navigation ECU is compromised via cellular interface* can be classified as technique t_{11} *exploit radio interface* (cf. ID 11 in Table A.4⁴), while formulating a_1 as *Navigation ECU*

⁴ derived from ATM-T0012 <https://atm.automotiveisac.com/technique/Exploitviaradiointerface>, accessed on Aug. 28, 2024

compromised would be too coarse to classify. Formally, to each attack step a_n , at least one technique $t_j \in \mathcal{T}$ is assigned. Classification of all steps of an attack path $\pi_p \in \Pi$ results in the technique-based attack path $\pi_{\mathcal{T},p}$ (cf. *IV* in Figure 8.4 and Figure 8.3), a finite sequence of techniques:

$$\begin{aligned}\pi_{\mathcal{T},p} = T(\pi_p) &= (T(a_1), \dots, T(a_n), \dots, T(a_{N_p})) \\ &= (t_1, \dots, t_j, \dots, t_{N_p}),\end{aligned}\tag{8.7}$$

where $t_1, \dots, t_{N_p}$ are arbitrary techniques $t_j \in \mathcal{T}$. The TTP catalog does not mandate a strict sequence of tactics to describe attack paths $\pi_{\mathcal{T},p}$, i.e., Techniques t_j, t_{j+1} are not necessarily used in sequential tactics θ_i, θ_{i+1} . They may be used in the same tactic, in tactics in reverse order, or they may be the same technique with different targets.

8.1.3 Risk treatment decision and derivation of monitoring goals (*II-d*, *II-e*, *III*)

Extending the risk treatment decision (*II-d*) in Figure 8.3) yields monitoring goals in addition to cybersecurity goals (cf. *III* in Figure 8.2, 8.3 and 8.4). To this end, *observability increase* (*II-e*) can be chosen alongside risk reduction. Observability increase does not contradict risk reduction: If reducing risks is appropriate (e.g., risk value of three or higher), increasing the observability of the threat scenario should also be considered. Conversely, low risks may not require preventative measures, but could still be observed (cf. Figure 8.3).

Defining monitoring goals has two benefits. First, as high-level requirements, they follow the same hierarchy and development process as preventive cybersecurity requirements. Second, differing between monitoring and cybersecurity goals improves the clarity of requirements, as it differentiates detection (monitoring) from prevention (cybersecurity). For example, if the threat scenario is *Message to the ECU was spoofed*, the cybersecurity goal *message to the ECU should be authenticated* is defined. The corresponding cybersecurity monitoring goal could be *unauthentic messages to the ECU shall be observed*. Since benign

actions may lead to the same observations as malicious ones, monitoring only the threat scenario may not be sufficient to observe an attack path. Hence, this work suggests specifying monitoring goals for several steps in an attack path. However, to balance observability and costs, considering the risk value of particular attack paths is necessary. This contrasts to ISO/SAE 21434, where risk values are only calculated per threat scenario, not per attack path.

Besides the attack feasibility rating previously estimated for each path, determining path-dedicated risk values requires a path-specific impact rating. This work proposes using the highest impact rating of all associated damage scenarios. A monitoring matrix (cf. Table 8.1) suggests the monitoring goals depending on the progression of the attack path (cf. root, penultimate, intermediate, leaf, Section *Example: HEAVENS 2.0 workflow* on page 47) and its risk value (cf. Remark 2.4). The matrix uses the attack tree-based nomenclature (cf. Figure 2.13 on page 47) to distinguish different steps in the attack path, including threat and damage scenarios. If the paths' risk value is low, only the threat scenario is monitored (risk value 2). For higher risks, monitoring goals for the entry points (value 3 and 4), and penultimate steps of an attack path (value 4) are specified. For very high risks (risk value 5), all attack steps, including intermediate steps, should be observable. Monitoring of damage scenarios is limited, but monitoring goals are

Table 8.1: Monitoring goal matrix. "x" indicates to define a monitoring goal for the given attack step. (cf. nomenclature of positions, Figure 2.13 on page 47)

Risk value	Position of step in attack path				
	Leaf	Intermediate	Penultimate	Threat scenario	Damage scenario
1					
2				x	
3	x			x	
4	x		x	x	x
5	x	x	x	x	x

adequate in case of higher risks (value 4 and 5). For example, observation of financial damage or loss of reputation is impossible in the vehicle. However, effects on safety and operability can be observed, such as status or crash information.

8.2 Monitoring concept definition

The result of TARA with threat information are attack paths described by techniques (cf. *IV* in Figure 8.4), and monitoring goals for specific attack steps (cf. *III* in Figure 8.4). This is the prerequisite to define monitoring requirements, allocated to logical components in a monitoring concept (cf. *VI* in Figure 8.4). This step also uses threat information, to map the techniques to potential observables (cf. *I-b*) in Figure 8.4).

Placing CMS has to consider their vantage on particular attack steps (cf. Section 5.2.1), depending on the architecture. For example, a network-based CMS will not observe traces of a privilege escalation on a host. Therefore, monitoring concept definition (*V* in Figure 8.4) considers the allocation of the requirements $r \in \mathcal{R}$ to the item or component(s) $\gamma \in \Gamma$.

8.2.1 Identification of relevant observables and observations (*V-a*)

To define the monitoring concept, it is first necessary to introduce the terms *observable* and *observation* and their relationship to attack steps and techniques. The monitoring concept should be able to provide all relevant observations, which is discussed in the following sections. Observables represent information that can be detected or collected by sensors.

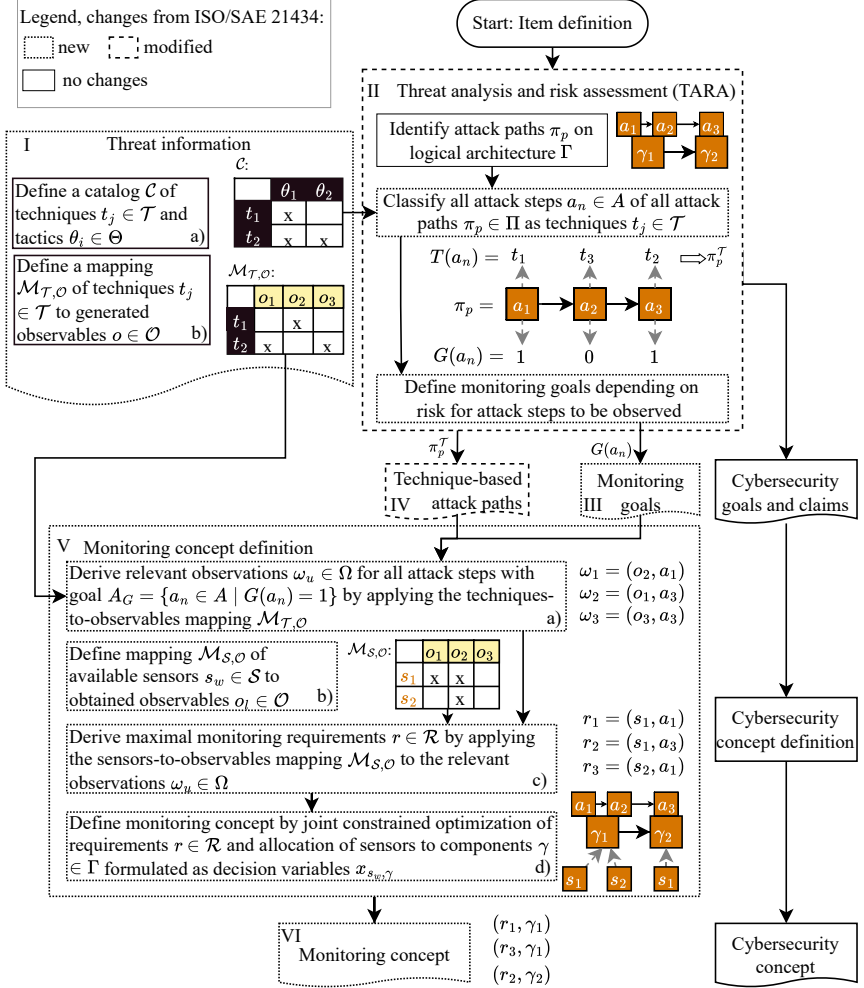


Figure 8.4: Visual overview of the entire procedure for specifying monitoring requirements, including modified TARA and monitoring concept definition. The color code equals the definitions in Table 5.2.

Definition 8.1: Observable

Cybersecurity information that can be obtained by sensors and is relevant to detect a particular technique or sub-technique. (derived from MITRE ATT&CK®⁵)

Observables o_l represent specific data points where the execution of a technique may introduce a measurable deviation from the system's normal or expected behavior. The relationship to techniques is formalized through the mapping $\mathcal{M}_{\mathcal{T},\mathcal{O}}$ that associates techniques $t_j \in \mathcal{T}$ with observables $o_l \in \mathcal{O}, l \in [1, L]$ (cf. I-b) in Figure 8.4), which is part of the required threat information for the development procedure. Classifying each attack step as a technique, and the known or defined observables generated by each technique $\mathcal{M}_{\mathcal{T},\mathcal{O}}$, enables the association of attack steps with observables.

Definition 8.2: Observation

An observation $\omega_u = (o_l, a_n)$ is defined as the association of an observable o_l with a specific attack step a_n .

The relevant set of observations $\omega_u \in \Omega, u \in [1, U]$ includes those associated with attack steps requiring monitoring, i.e., attack steps for which a monitoring goal is defined ($G(a_n) = 1$). The set of attack steps subject to monitoring is defined as

$$A_G = \{a_n \in A | G(a_n) = 1\}. \quad (8.8)$$

Consequently, using the attack steps classified as techniques $T(a_n) = t_j$ and the mapping of techniques to observables $\mathcal{M}_{\mathcal{T},\mathcal{O}}$, the set of relevant observations is given by:

$$\begin{aligned} \Omega = \{ & (o_l, a_n) \in \mathcal{O} \times A_G \mid \\ & \forall a_n \in A_G, T(a_n) = t_j \\ & \text{such that } (t_j, o_l) \in \mathcal{M}_{\mathcal{T},\mathcal{O}} \} \end{aligned} \quad (8.9)$$

⁵ <https://attack.mitre.org/datasources/>, accessed on 2025-02-14.

In the shown procedure (cf. Figure 8.4), step a_1 is classified as technique t_1 , which can result in generation of observable o_2 . An observation $\omega_1 = (o_2, a_1)$ monitors the attack step by collecting the relevant observable. For example, if t_1 is *exploit radio interface*, a potential observable o_2 may be an *application log* (cf. ID 6 in Table A.5), that shows an unstable behavior of the system.

Whether a technique generates a particular observable depends on the implementation of the logical architecture by the E/E architecture. Determining the relevant observations should therefore consider whether the logical component is likely to be realized on a POSIXTM-based (e.g. AAP), or a ACP host (cf. Section 2.1.2 on page 18), and whether the technique involves an Ethernet network or a bus. For example, logs of the OS cannot be collected on a host without an OS, and spoofing of CAN messages cannot be detected on the bus because the messages do not identify the sender.

Note, that the mapping $\mathcal{M}_{\mathcal{T},\mathcal{O}}$ (I-b) identifies observables where deviations from normal behavior *may* manifest. However, observables can also reflect normal system behavior. For example, permitted applications start processes to execute code⁶, but processes could also be started by malicious code. Therefore, the presence of an observable is not definitive proof of a certain technique, but rather an indicator of its potential occurrence.

8.2.2 Mapping the capabilities of available sensors (V-b)

The considered CMS realizations⁷ $s_w \in \mathcal{S}, w \in [1, W]$ are analyzed. The capabilities of the available sensors $s_w \in \mathcal{S}$ are determined by identifying the set of observables $o_l \in \mathcal{O}$ each sensor can detect. This relationship is formalized through the sensor capability mapping $\mathcal{M}_{\mathcal{S},\mathcal{O}} \subseteq \mathcal{S} \times \mathcal{O}$. A pair (s_w, o_l) belongs to the mapping $\mathcal{M}_{\mathcal{S},\mathcal{O}}$, if the sensor s_w can collect or detect the observable o_l . For example, if a specific *network intrusion detection system* sensor $s_{NIDS} \in$

⁶ on hosts with a POSIXTM-based OS

⁷ e.g., IDSs of different suppliers

$\mathcal{S}$ can detect the observable o_1 *network traffic* (cf. ID 1 in Table A.5), then $(s_{NIDS}, o_1) \in \mathcal{M}_{\mathcal{S}, \mathcal{O}}$.

8.2.3 Establishing the maximal monitoring requirements (V-c)

Building upon the sensor capability mapping $\mathcal{M}_{\mathcal{S}, \mathcal{O}}$, the specific monitoring requirements for attack steps to be observed $a_n \in A_G$ are established. A monitoring requirement is represented as a pair (s_w, a_n) , signifying that sensor $s_w \in \mathcal{S}$ is suitable for observing attack step a_n . Suitability holds if the attack step a_n is associated with an observable o_l (i.e., $\exists(o_l, a_n) \in \Omega$) that the sensor s_w can detect (i.e., $(s_w, o_l) \in \mathcal{M}_{\mathcal{S}, \mathcal{O}}$). The set $\mathcal{R}$ comprises all such valid requirements:

$$\mathcal{R} = \{(s_w, a_n) \in \mathcal{S} \times A_G \mid \exists o_l \in \mathcal{O} \text{ such that} \quad (8.10)$$

$$(o_l, a_n) \in \Omega \text{ and } (s_w, o_l) \in \mathcal{M}_{\mathcal{S}, \mathcal{O}}\}$$

This set $\mathcal{R}$ represents the maximal monitoring requirements, as it includes all valid sensor-attack step pairings derived from potentially multiple observables per attack step and the various sensors capable of detecting them. For example (cf. Figure 8.4), the attack step a_1 *Navigation ECU is compromised via cellular interface*, classified as t_j *exploit via radio interface*, may generate the observables o_l *application log* (application instability), and o_m *network traffic* (external communication). The sensor s_1 *network intrusion detection system* may collect o_m , while the sensor s_2 *log collection system* may collect o_l . Accordingly, maximal requirements are $\mathcal{R} = \{r_1, r_2\}$ with $r_1 = (s_1, a_1)$ and $r_2 = (s_2, a_1)$, observing one attack step by collecting two observables with two sensors. It is therefore not necessary to fulfill *all* requirements to monitor all attack steps that shall be observed.

8.2.4 Optimization and allocation of requirements (V-d)

After determining the maximum requirements, i.e. all suitable sensors to observe the relevant attack steps (V-c) in Figure 8.4), a (sub)set of requirements must be selected. Since some attack steps may be covered by multiple sensors, the requirements can be optimized by selecting a minimal amount of sensors to observe all attack steps. The allocation of sensors to components γ should be combined with the optimization, because the allocation determines the sensors' vantage. The combined optimization and allocation can be formulated as a joint optimization problem with the binary decision variable $x_{s_w, \gamma}$:

$$x_{s_w, \gamma} = \begin{cases} 1 & \text{if sensor } s_w \in \mathcal{S} \text{ is allocated to component } \gamma \in \Gamma \\ 0 & \text{else} \end{cases} \quad (8.11)$$

The constraint of the optimization is that all attack steps A_G can be observed by at least one sensor:

$$\sum_{s \in \mathcal{S}, \gamma \in \Gamma, (s, a) \in \mathcal{R}} x_{s, \gamma} \geq 1, \forall a \in A_G \quad (8.12)$$

The optimization and allocation is then formulated as a minimization problem with

$$\min \sum_{s \in \mathcal{S}, \gamma \in \Gamma} x_{s, \gamma} \quad (8.13)$$

if costs are not considered, i.e. just the number of allocated sensors is minimized. For example, to observe the attack step a_1 *Navigation ECU is compromised via cellular interface*, just the sensor s_2 *log collection system* may be used, and allocated to the navigation ECU.

The optimization can include other factors, especially if the deployment costs (e.g. CPU load) are to be taken into account. For example, a cost-benefit assessment can be carried out (development costs, hardware costs, bus load, etc.), and redundancy

can be considered (e.g. host domain and network domain). In this case, the minimization is

$$\min \sum_{s \in \mathcal{S}, \gamma \in \Gamma} f(x_{s,\gamma}, s, \gamma) \quad (8.14)$$

with a cost function f defining the costs of deploying s to γ . Moreover, fine-grained evaluations of observability could enable further optimizations. Parameters such as the time window of observation, the discriminative power of the information (benign versus malicious), or the skills required to hide the technique from observation could be suitable to trade-off between different solutions.

9 Holistic cybersecurity monitoring of vehicle fleets

After detailing the procedural approaches of this work in the chapters 7 and 8, this chapter presents the technological approach to the ideas ③, ④, and ⑤ in depth. Overall, the approach is referred to as *holistic monitoring* and expands upon the introduced concept in chapter 6 (cf. Figure 6.4).

Remark 9.1: Holistic monitoring

Holistic monitoring combines in-vehicle monitoring of cyber and physical aspects with OEM-internal and external sources, enabling comprehension of the cybersecurity situation in the VSOC.

Different areas contribute to holistic monitoring (cf. Figure 9.1): the vehicle fleet (cf. Section 9.1.1), the development process (cf. Section 9.1.2), the support organization (cf. Section 9.1.3), and the operations process (cf. Section 9.1.4). In the vehicles (cf. lower half of Figure 9.1), suitable monitoring sensors observe the cyber and physical aspects. The presented development procedure (cf. Chapter 8) specifies the sensor requirements. After collection in the VSOC, a KG (cf. Figure 9.1) links the observations from the vehicles with further knowledge, including development process artifacts (cf. upper left of Figure 9.1) to relate collected observables from the fleet to configuration, components, threats, and risks. The support organization is the point of contact for external sources, such as white hat hackers and customers. With that, reports on potential problems are added to the KG (cf. center left in Figure 9.1).

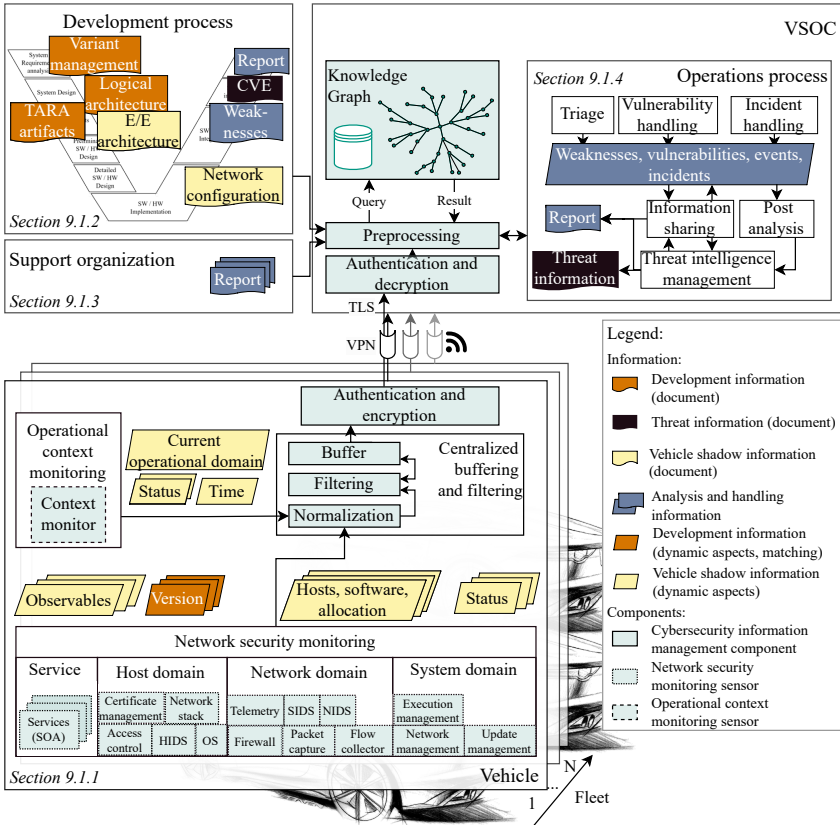


Figure 9.1: Overview of the design of holistic monitoring (refines Figure 6.4), combining in-vehicle monitoring, development process information, the support organization and the operations process. The color code equals the definitions in Table 5.2.

The operations process (cf. Chapter 7) is the main user of the knowledge in the KG but also feeds in further information (cf. upper right in Figure 9.1). In particular, threat intelligence management yields CTI (from information sharing and internal continuous improvement) to describe attackers and their methods.

At the same time, the triage, vulnerability, and incident handling phases use all data in the KG to identify weaknesses, vulnerabilities, events, and incidents.¹

The following section outlines the information collected from vehicle fleet, development process, support organization, and operating process, and describes the considered views (cf. Section 9.1). The detailed design of the approach in the vehicles (cf. Figure 3.1 on page 57) is then presented (cf. Section 9.2, addressing idea ③), including service domain monitoring for SDVs and taxonomies of cybersecurity monitoring sensors and operational context. For fleet-wide collection, the information is buffered in the vehicle and transmitted securely via VPN and two-way authentication (cf. Section 9.3, addressing idea ④). Finally, the KG within the VSOC (cf. Section 9.4, addressing idea ⑤) fuses all information. In particular, its realization as LPG provides benefits for managing cybersecurity information (cf. Section 9.4.1), while the proposed data model enables the semantic linking of the four views under consideration.

9.1 Overview of the information considered in holistic monitoring

The VSOC is the core of cybersecurity operations for all vehicles of an OEM, with the proposed operations process (cf. Chapter 7) as its backbone. In the monitoring phase of the operations process (cf. Section 7.2.2), the holistic monitoring approach integrates in-vehicle information, development documents, the support organization, and the operations process to enhance the Cybersecurity Situation Awareness of the VSOC. Each area provides specific information to complete the four views of the VSOC on the cybersecurity situation. The data model (cf.

¹ The following chapter was published previously in parts by this work's author [Gri+24c]: D. Grimm, M. Zink, M. Schindewolf and E. Sax, "Cyber Situational Awareness in Vehicle Security Operations: Holistic Monitoring and a Data Model," 2024 20th International Conference on Network and Service Management (CNSM), Prague, Czech Republic, 2024, pp. 1-7, doi: 10.23919/CNSM62983.2024.10814319. ©2024 IEEE.

Section 9.4.1) links all considered information (cf. detailed specification in the appendix Table A.7).

Table 9.1: Information sources for holistic monitoring and their contribution to the four views.

Source	Views fed	Information
Vehicle fleet	 Vehicle shadow	Observables, network addresses, software and hardware instances, operational context
	 Development	Current software and hardware versions, active configuration
Development process	 Development	Variants, logical architecture, TARA artifacts
	 Vehicle shadow	Static E/E architecture configuration
	 Threat information	References to external vulnerability and threat databases
	 Analysis and handling	Weaknesses and reports from source code scanning and testing
Support organization	 Analysis and handling	Unverified reports from bug bounty, responsible disclosure, field returns, ecosystem
Operations process	 Threat information	Threat actors, techniques, tactics, malware, tools, indicators
	 Analysis and handling	Weaknesses, vulnerabilities, incidents, courses of action, unverified reports from external sources

9.1.1 Vehicle fleet

The vehicle fleet contributes to two views: the vehicle shadow view with runtime data such as observables and operational context, and supports the mapping to the development view by providing the current configuration and system state required to identify the corresponding development artifacts (cf. Table 9.1 and lower half of Figure 9.1). The vehicle shadow mirrors the actual vehicles, as such, there is a vehicle shadow per produced vehicle. Whether vehicle shadow information originates from the vehicle or development documents (cf. Section 9.1.2) depends on SDV software dynamics. With increasing software dynamics, in particular

containerization and orchestration [Sch+22; Sto+21], parts of the considered information on the E/E architecture are established at runtime by the vehicle.

Vehicle shadow The vehicle shadow unites observables from the vehicles, a representation of the network where the observables were generated (i.e., the E/E architecture), and the operational context to relate information to the vehicle's situation. The vehicle shadow represents instance-level realizations of vehicles, which relate to the abstract, modeled elements in the development view where applicable.

Cybersecurity monitoring sensors are the primary source of observables. Different sensors generate manifold types of observables (cf. Section *Cybersecurity monitoring sensors and observables* on page 176 and Table A.5). For instance, AUTOSAR standardizes logging, IDS, access control, and firewalls, providing application logs, network traffic, and file accesses. In SDVs, where the network configuration is partly dynamic, the vehicle may also inform about network addresses via the orchestrator.

Operational context monitoring focuses on the physical situation of the vehicle as defined by the ODC [ISO23a] (cf. Definition 2.18), including the environment, motion status, and status of functions and components. This thesis proposes a centralized component collecting the operational context (cf. Section 9.2.2). Beyond the COD, errors and general status information (such as DTC²) may be collected via diagnostic interfaces, AUTOSAR mode, and network management.

The vehicle shadow is completed by an accurate and comprehensive representation of the vehicle's network. Traditionally, E/E architectures remained largely static after development. While development artifacts contain the static aspects of the vehicle shadow, in SDVs, information on the variable software deployment at runtime must be provided by the vehicles. Regarding the software allocation to hosts and virtual hosts, the execution management provides information via an API. Furthermore, the update management service can identify the physical hosts

² Diagnostic Trouble Codes

and software components, in particular ACP hosts. Moreover, IT tools [LN24] or automated methods [MP21] can dynamically identify networks and services. However, static information available in development documents should not be extracted from customer vehicles unnecessarily, the focus should be on dynamic aspects, particularly software allocation.

Development Tightly connected with the software allocation is the challenge of identifying the vehicle's current system revision and configuration, which is important to identify vulnerable vehicles. Due to repairs and software updates, the actual versions of software and hardware components are only known in the vehicles. Initial hardware versions are known at production, but repairs may change them over the vehicle lifetime, and OTA updates frequently change the installed software. Moreover, software update delivery varies across vehicles due to connectivity issues or customer preferences. To connect the vehicle shadow to the correct components and configuration in the development view, the current software and hardware versions must be fetched from the vehicles. The update management service, workshop interfaces (UDS, OBD), and the OS³ are the key sources for current version and active configuration information.

9.1.2 Development process

The development process (cf. upper left in Figure 9.1) primarily feeds the development view from models and documents, while also contributing to threat information and analysis and handling via automated security analyses and testing. Tool-supported development is particularly relevant, as digital artifacts may be available in this case, and software can automatically parse and insert the information into the KG (cf. preprocessing in Figure 9.1, upper center). Beyond, automated tests in CI/CD systems can provide further artifacts, especially for the threat information, and analysis and handling views. Artifacts generated during

³ POSIX™ OS allow to query their version at runtime

development are transferred to the KG as part of the *preparation* phase of cybersecurity operations (cf. Section 7.2.1) and continuously updated throughout the lifecycle, e.g. due to a software update.

Development In the development view, information from variant management, logical architecture modeling, and risk management are fused. Variant information focuses on SPLE-based models, including features, configurations, system revisions and system generations. For modeling the architecture, an SOA-based (cf. Definition 2.1) approach is assumed, which considers components as services with particular capabilities. The development view is completed with information regarding risk management, focusing on TARA artifacts as defined in ISO/SAE 21434 [ISO21], aligned with the proposed threat information mapping (cf. Chapter 8).

Various tools used throughout the development process produce digital artifacts used to build the development view in the KG. From variant management, AUTOSAR supports feature models in the form of XML⁴ files for configurations [AUT21], and PREEvision [Vec24] describes product lines with feature models. The variant selected by the customer determines the installed hardware during production. As a result, a vehicle's built-in features are digitally available. If customers can purchase vehicle features on-demand, the OEM has to change the installed features in a particular vehicle, requiring continuous synchronization of the development view and vehicle shadow with a database of on-demand purchases.

For modeling logical and E/E system architectures, tools such as PREEvision and System Composer [Mat24] are used in the industry. While these tools use different nomenclatures, e.g., based on AUTOSAR, in the following, the SOA (cf. Definition 2.1) terminology is used, which considers components as services with particular capabilities. Even if the logical architecture is not fully SOA-modeled, the development process provides other models suitable to derive dependencies

⁴ Extensible Markup Language

and capabilities, such as component interfaces. Interface specification formats like AUTOSAR's XML files or ROS 2 message files enable automated identification of capabilities, and tools export the versions of basic software (e.g., version of ACP⁵).

Risk management information focuses on TARA artifacts, which may require manual input to the KG if it is unstructured (e.g., from brainstorming). The collaboration with suppliers requires managing and exchanging TARA, which pushes the development of formats like openXSAM [ite23] and tool support for TARA. Accordingly, digital TARA artifacts are expected to become common, and the proposed TARA management process can provide access to them (cf. Section 7.3.1). In addition, comparing development estimates with actual attacks (cf. Section *TARA management as a continual activity* on page 145) can improve TARA artifacts over time.

Vehicle shadow Development artifacts define the static aspects of the vehicle shadow, instantiated for each vehicle post-production. All static or rarely changing aspects (e.g., installed hardware) of the vehicle network are defined during the development process and then replicated in the KG for each produced vehicle. Furthermore, the corresponding elements of the vehicle shadow can be linked to their abstractions in the development view (e.g., running software in the vehicle relates to a modeled component). The standard AUTOSAR-based development tools export information on network topology, hardware, bus systems, and software allocation in the XML format, for example. Moreover, the development process can provide information about the static aspects of the network configuration, including static IP addresses, MAC addresses, host names, domain names, message ID⁶ on CAN buses, and certificates used for authentication.

⁵ cf. page 20

⁶ identifiers

Threat information Automated build systems for CI/CD processes often integrate vulnerability database scanning tools, that generate SBOMs (cf. Remark 4.4). SBOMs primarily contribute structural information to the development view (software components, including contained third-party components), while also enabling linkage to threat information through vulnerability database matching. Thus, the threat information view references CVE entries for each contained software component with known vulnerabilities.

Analysis and handling For every detected vulnerability in a third-party software component using vulnerability database scanning, the preprocessing can preliminarily assign a weakness, which is then subject to further analysis in the operations process. Source code scanning tools do not reference specific vulnerabilities from public databases. However, they can be used to identify weaknesses in a company's proprietary software components, contributing to the analysis and handling view. The development process should repeat vulnerability database and source code scans when the public databases or scanning systems are updated, including for vehicles produced in the past, to detect newly discovered vulnerabilities in software already in the field.

Moreover, the development process executes cybersecurity tests, some of which are also carried out automatically in CI/CD systems. The results are initially represented as (unstructured) reports in the analysis and handling view, where unverified findings are stored before being linked to confirmed vulnerabilities or incidents. For example, fuzz testing uses random inputs to identify bugs in software that can potentially lead to vulnerabilities. The statistical nature of fuzz testing may lead to deferred discovery of bugs when the software is already in use. Also, due to improved testing, bugs might be found after production when already relevant to operations.

9.1.3 Support organization

Analysis and handling The support organization (cf. Figure 9.1, center left) contributes to the analysis and handling view in the form of reports. The considered sources are the bug bounty program, field return or quality reports received by the dealers, and information from the ecosystem (telecommunication providers, etc.). However, these sources do not directly provide validated entities such as weaknesses or events, but unstructured information that still needs to be checked for correctness and relevance. In contrast to threat information, which represents generalized knowledge, these reports are OEM- or problem-specific and operational in nature. ISO/SAE 21434 strongly requires external sources, which makes reports a common source. In particular, this includes external sources without a contractual relationship of trust, so a preliminary check of the information is essential. After verification, the information contained in reports can be linked to or transformed into validated entities such as vulnerabilities, weaknesses, or incidents.

9.1.4 Operations process

Unlike the automated preprocessing of development artifacts or observables from the fleet, the operations process involves manually inserting data into the graph in iterative tasks through the VSOC personnel. In particular, during triage and incident analysis, a human operator creates queries and the information to insert. Therefore, the operations process is the primary user of the data in the KG, querying the database for correlation and writing new information to track the history of incidents and vulnerabilities. The operations process contributes primarily to the analysis and handling view, but also the threat information view.

Threat information The threat information view relies upon external sources, analyzed within threat intelligence management by a particular role in the VSOC, the threat intelligence analyst. External automotive CTI sources include open sources like databases and catalogs (cf. Section *External sources* on page 80),

deep and dark web forums and marketplaces, and closed communities, including commercial platforms. Internal knowledge from past analyses, e.g., attacker techniques and tools, should also be formalized within the *post-analysis* phase to improve threat intelligence (cf. Section 7.3.4). The VSOC's responsible role for information sharing collaborates with threat-sharing organizations like Auto-ISAC [SC24], which may provide further threat information.

Analysis and handling Within the operations process, all phases contribute to the analysis and handling view. The triage phase derives indicators from observables (from the vehicle shadow) and reports, establishing relationships to other indicators through algorithms or manual input. The escalation of security events to incidents, or weaknesses to vulnerabilities, can update these connections. As remote access is uncommon, response execution should be automated (cf. Chapter 10), similar to deploying software updates. As a result, courses of action can be automatically proposed or inserted be automatically in the KG based on predefined response strategies.

Beyond the triage, analysis and response tasks, the threat intelligence management and the supplier interactions contribute to the analysis and handling view as part of the operations process. Threat intelligence management contributes to the analysis and handling view when threat intelligence is contextualized to specific assets or vehicles. For example, a finding about a hacker selling an exploit for a specific vehicle model in deep web forums may involve a vulnerability. Textual inputs from such external sources, represented as reports in the analysis and handling view, may initially be unassociated, but become relevant for the analysis and handling view once an analyst links them to concrete assets. While the threat information view contains generalized, external knowledge about threats, the analysis and handling view captures OEM-specific operational records derived from this information.

Suppliers can also contribute directly to the analysis and handling view through information sharing if the OEM has established explicit rules for exchanging information about vulnerabilities and incidents. If software or hardware from

suppliers is already in the field, information about potential problems is highly relevant to secure the vehicles on the road. Integrating structured information from suppliers benefits from the inherent trust in the correctness of the information due to the established contractual partnership and ease of sharing confidential details about root causes of vulnerabilities.

9.2 In-vehicle monitoring of cyber and physical aspects ③

9.2.1 Network security monitoring

Cybersecurity monitoring sensors and observables

Based on the literature, including considering AUTOSAR's monitoring events, a taxonomy was developed (cf. Figure 9.2), which categorizes monitoring sensors commonly found in vehicles. The taxonomy extends Collins' domain model [Col14] with the *system* domain, which includes components required for the operability of the vehicle network.

The network domain contains NIDS, firewall, flow collectors, and packet capture. In addition, particular signals are the object of monitoring within SIDS⁷ and telemetry systems. Logs of OS, access control, and HIDS are part of the host domain. The sensors *certificate management* and *network stack* are derived from AUTOSAR events. On the service domain, arbitrary SOA services are the sensors, which generate logs (cf. Section *Monitoring the service domain of software-defined vehicles* on page 177) and status information. In the system domain, network management services, the update management service, the execution management for SDVs, and cybersecurity information management are considered. Network management sensors include DHCP servers, DNS forwarders, and

⁷ Signal-based Intrusion Detection System (cf. glossary, page 329)

AUTOSAR network management, which can switch hosts on and off to save energy. Execution management includes the SDV specific services such as container runtime, orchestrator, and VM hypervisor. Through their authorizations to control the software, they can provide various observables and additional information on the E/E and logical architecture. The CIM can be used as a cybersecurity sensor (cf. Definition 4.2 on page 73) when it reports errors and modifications. The most vital cybersecurity "sensor" in the system domain is the update management service. It observes diagnostic sessions, (failed) authentication, and updates or diagnostic commands to change software. Moreover, it informs about the versions of software and hardware and can provide status information, such as DTC and running hosts.

A set of common observables was derived from the literature. The proposed definitions of observables (cf. appendix, Table A.5) retain ATT&CK®'s terminology as far as possible. However, particular static (e.g., certificate modification, resource utilization) and real-time properties (e.g., o_2 : Clock) caused the necessity for some automotive-specific observables.

In general, the collected observables depend on the specific implementations of a sensor. For example, while a particular HIDS may observe the change of files (cf. o_{11} in Table A.5), the HIDS of another manufacturer cannot. Nevertheless, the different domains of the sensors implicate some general differences, allowing the allocation of common observables to sensor types. For example, sensors in the network domain mainly monitor network traffic (o_1) but can also detect new hardware in the network (o_3 : Asset), except for signal-based sensors, that focus on particular information (o_7 : Measurement). OS logs and HIDS have a large overlap of generated observables, as HIDS (depending on the OS) access the OS logs.

Monitoring the service domain of software-defined vehicles

Containers and SOA are the expected technologies to implement applications in SDVs. However, collecting logs of these applications poses challenges, as

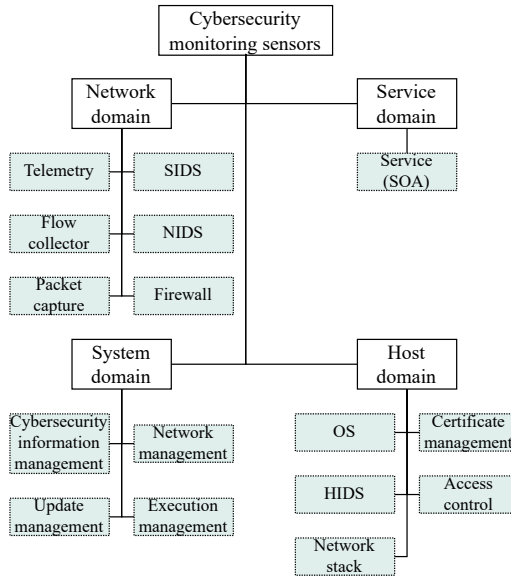


Figure 9.2: Taxonomy of cybersecurity monitoring sensors.

automotive SOA applications vary by vehicle variant. Logging must cater to these unique applications. Currently, automotive application developers can log relevant information via middleware mechanisms such as DLT in AUTOSAR or the built-in logging of ROS 2. Yet, this data is only collected in specific cases for debugging during the development, but not during operations for cybersecurity purposes.

Based on that, the design of SDV service monitoring builds on the developers' knowledge of their applications about critical or important states and behaviors. The proposed approach aims to provide a simple way to log information to developers. "Simple" in this context means that the developer does not need additional knowledge to contribute observables other than the knowledge of SOA technology to develop the SDV service.

Even on AAP based hosts, the POSIX™ basis allows the usage of containerization technology. To use the AAP's DLT mechanism with containerized applications,

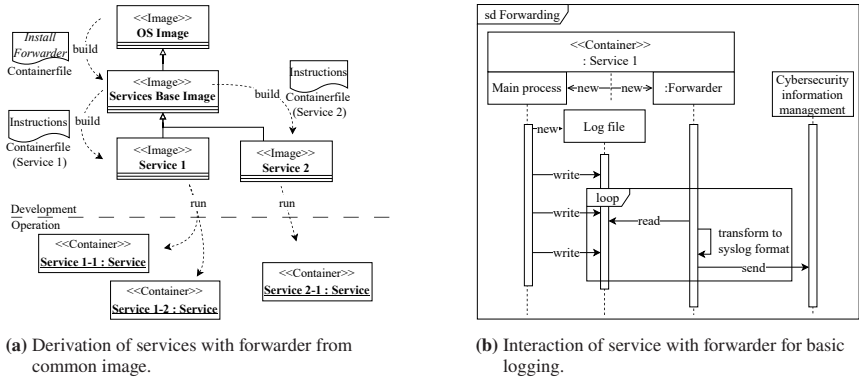


Figure 9.3: Service domain monitoring by equipping containerized services with a forwarder for basic logging.

the file system of the host could be mounted in the container [Suf22] so that the logged data can be processed further on the host by the DLT daemon⁸. However, a different approach that focuses on the flexibility of containerized services is proposed. A service should be as independent as possible from the executing platform to achieve maximum flexibility in deployment. Accordingly, the logging functionality is fully integrated into the containers.

As a result, all services are equipped automatically with *basic logging* functionality. By leveraging the containerized SDV development workflow, all services derived from a shared basic container image can make use of the pre-installed *Forwarder* (cf. Figure 9.3a). The forwarder's task is to read the application's logs within the container, transform them to a joint protocol (e.g., syslog), and send them to the centralized buffering and filtering component (cf. Figure 9.3b and case *a*) in Figure 9.6). The realization of the forwarder depends on the SOA middleware. For AAP, for example, a modified DLT daemon could provide the necessary interface to the application.

Besides, *extended logging* allows the collection of any additional structured information. Developers can use a dedicated SOA interface that mimics the selected

⁸ continuously running process that accepts log messages and specific control commands

joint protocol (cf. case *b*) in Figure 9.6) to forward data they deem relevant. Thus, developers can provide purpose-built information using the conventional publish-subscribe mechanism. For example, in the prototypical implementation of this thesis, a ROS 2 interface for the syslog protocol was developed. The extended logging mechanism enables using the same fields and constraints within the SOA as defined in the joint protocol, but without knowing its particularities. As a result, the specification of detailed requirements for cybersecurity monitoring for each SOA application is possible without requiring additional technology in the services.

9.2.2 Operational context monitoring

Operational context monitoring complements network security monitoring by capturing the vehicle's physical situation. While network security monitoring focuses on observance within the digital realm, context monitoring enhances this by integrating data from a sensory and functional perspective.

Operational context of vehicles

The term *context* is defined as "any information that can be used to characterize the situation of an entity [...]" [DA+00], leading to numerous ways to define the (operational) context of a vehicle. The proposed approach builds on two thoughts, focusing on the vehicle's cybersecurity situation. Firstly, the relevance of context information depends on a vehicle's specific cybersecurity risks, which are determined by the architecture and identified in the TARA. ISO/SAE 21434 encourages this thought, as it mandates defining the operating conditions under which identified threats are relevant. Secondly, information related to the (potentially automated) driving functions is crucial. If a safety-critical situation occurs, it is essential to ensure that this is not related to a cybersecurity problem. Therefore, safety-relevant information is indirectly relevant to cybersecurity.

However, due to the high number of signals, their volume, and the need for wireless transmission to the backend, it would neither be economically viable nor technically feasible to collect all raw signals that are potentially relevant for safety and security. Hence, the proposed approach considers the operational context as a summary of the situation and sends the collected information to VSOC only when appropriate (not every time the conditions change). To this end, one could define a custom set of abstracted, summarized information capturing the vehicle's operational environment. However, resorting to established safety-related taxonomies is beneficial for limiting the additional efforts in development and operations to observe the operational context. In addition, this ensures consistency between the safety context and the cybersecurity context.

Because the ODD and the associated ODC (cf. Section) encode the nominal context(s) of safety-critical functions, they are the basis for designing the context monitoring approach. As outlined in the ISO 34503 [ISO23a] and ISO 34504 [ISO24b] standards, the primary categories of information to define an ODD are environmental conditions, scenery, and dynamic elements. The ODC (cf. ISO 34503 [ISO23a]) adds status (including motion status and status of components) and user status to this list. User status is mentioned mainly for the sake of completeness when describing the operational context, but has little direct influence on cybersecurity. By contrast, wireless connectivity is particularly relevant for external attacks, falling under environmental conditions. The scenery has an indirect influence on cybersecurity, as zones (e.g., countries) and drivable areas (highways, inner cities) are important and distinctive aspects to define the safe operational boundaries of functions. Accordingly, this information is important to assess which functions are currently allowed to influence the vehicle. The dynamic elements serve to further contextualize this information. Besides, this work adds a temporal dimension, distinguishing usual operating hours or nighttime, which is a key factor in IT security considerations (cf. Figure 9.4). The presumably most important aspect of the operational context is the status information. The activity of hosts, running services and ongoing updates provide the necessary context for analyzing incidents, as this data allows differentiation between planned processes, such as maintenance and updates, and potential

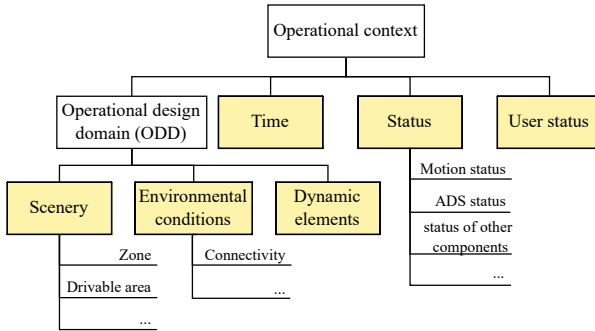


Figure 9.4: Elements of the operational context.

anomalous actions. In addition, the simple movement behavior (standstill or movement) is a straightforward and frequently used criterion to allow or block various critical actions such as updates or diagnostic functions. Accordingly, this information should also be found in a KG.

In conjunction, the operational environment of an item defined in the TARA should consist of the same set of information. If a threat scenario of an item is relevant in a certain status of a function of the item, then this status is part of the operational environment. The ODC definition should consider such statuses.

Collecting operational context

After specifying the operational context, collecting the information requires technical solutions. Considering that the ODD and ODC for highly automated driving functions must be specified, monitoring compliance with the ODD is essential. In particular, the current operational domain must be a subset of the defined domain ($COD \subseteq ODD$). The other elements of ODC, such as the status and behavior of the vehicle, must also be verified so that current conditions $\subseteq ODC$. Two approaches to monitor the COD can be distinguished: centralized and decentralized. In a decentralized approach (cf. Figure 9.5a), each in-vehicle component that has a defined ODD would monitor the fulfillment of its operational design conditions individually. However, incongruous ODDs of dependent components

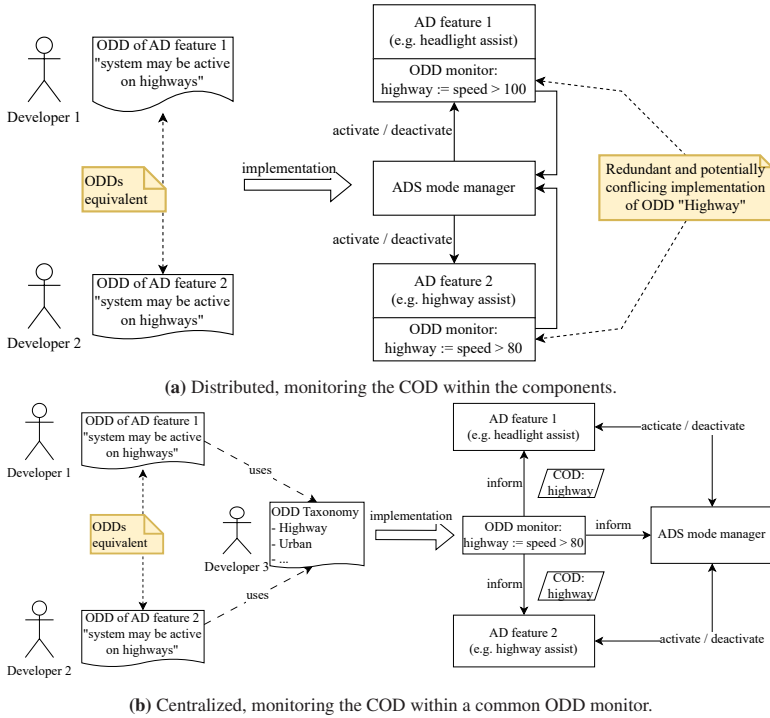


Figure 9.5: Schematic comparison of distributed versus centralized assurance of the ODD of automated driving functions.

(e.g., one is deactivated, but the other expects it to be active) could introduce misconfigurations.

Accordingly, the assumption is that a centralized component monitors the operational context, including COD and conditions (cf. Figure 9.5b). Hence, monitoring the currently "active" operational domains ensures synchronized activation and deactivation of functions. The ISO technical report 4804 [ISO20b] states that the mode management of ADS should be responsible for handling the ODD, supporting the idea of centralized monitoring.

For safety, the collected information controls the activation and deactivation of functions. For cybersecurity, centralized context monitoring allows control of the timing and amount of information forwarded to the VSOC. Control of timing involves periodically capturing the COD and during startup, safety-critical situations, activation or deactivation of automated driving functions, updates, and diagnostic accesses. The extended logging mechanism (cf. Section *Monitoring the service domain of software-defined vehicles* on page 177) is the interface to centralized buffering and filtering. The context monitor sends the structured⁹ information using the SOA interface.

9.3 Fleet-wide collection ④

Due to the implementation of the identified cybersecurity monitoring requirements, each vehicle features cybersecurity monitoring sensors and operational context monitoring. These components provide information aligned with the identified risks during vehicle operation. A centralized buffering and filtering CIM component (cf. Section 9.3.1) handles the resulting data in the vehicles. The CIM component integrates with the vehicle's cybersecurity monitoring sensors (cf. Figure 9.6) and provides a secure channel to the VSOC (cf. Section 9.3.2), to collect the data of the entire vehicle fleet.

9.3.1 Centralized buffering and filtering

Integration with cybersecurity monitoring sensors

There are five different cases when integrating cybersecurity monitoring sensors and context monitoring with the CIM (cf. a) to e) in Figure 9.6), depending on the available interfaces of the sensors. Cases a) and b) have been discussed

⁹ Multiple context classes such as time, environmental conditions, scenery, with a list of sub-elements

within the proposed approach to monitor the service domain in SDVs (cf. Section *Monitoring the service domain of software-defined vehicles* on page 177), whereby the extended logging interface *b*) is also used by the context monitor (cf. Section 9.2.2).

The sensor deployment in case *c*) is similar to case *a*), as both involve containerization, but the sensors cannot natively send data in the joint protocol. Hence, a forwarder is part of each container in both cases. In case *d*), a sensor is installed on a specific host and must also be equipped with a forwarder that converts the information produced into the joint format and forwards it. For example, collecting logs of the host OS could require such a procedure. Case *e*) shows the sensors that can already send data in the joint protocol.

Normalization and filtering

Continuously sending all data the deployed sensors produce to the VSOC would lead to unnecessary amounts of data being recorded, as the vast majority of the time, nothing critical happens. Therefore, to ensure efficiency, a filter functionality is proposed. The normalization merges the data from all sources (sensors and context) into a single data stream. A *log level* configured for individual vehicles determines what data is filtered.

The log level indicates the minimum severity of information to be forwarded. For example, in the syslog protocol, 0 is the highest, and 7 is the lowest severity. The severity of incoming messages is checked against this level, acting as a threshold. Messages with a severity equal to or higher than the log level (meaning they are at least as severe as specified) are forwarded to the output buffer. Messages with a lower severity (less critical) are written to the filter buffer. New messages override the oldest entries when the filter buffer is full.

The adaptation of the log level [Gri+23a] serves as a reaction possibility (cf. *f*) in Figure 9.6, detailed discussion in Chapter 10). After each change to the log level, the component iterates over the filter buffer once. All entries with a severity less

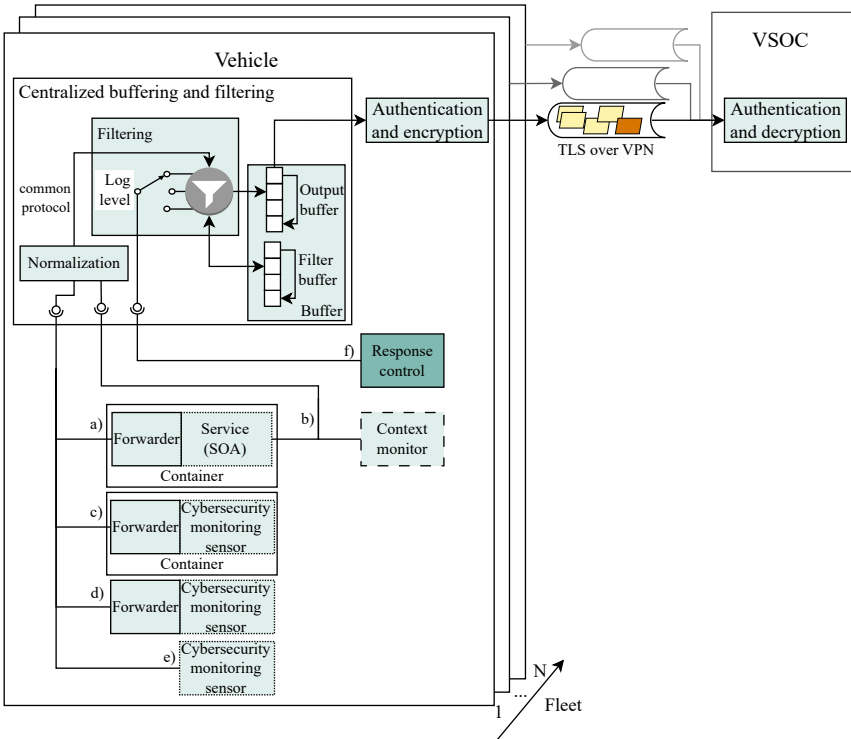


Figure 9.6: Cybersecurity information management in the vehicle: centralized buffering and filtering (details of secure transmission from vehicle to VSOC: cf. Figure 9.7).

than or equal to the new log level are removed from the filter buffer and forwarded to the output buffer.

Output buffer

The output buffer's task is to bridge times without connectivity to the backend. If the connection to the VSOC is established, the information in the output buffer is forwarded using the proposed secure channel (cf. Section 9.3.2). If there is no connection, the information to be sent is buffered. However, when the output

buffer is full, the oldest or least critical information has to be discarded. Informing the VSOC about the information loss would be appropriate.

9.3.2 Secure channel for monitoring data

A two-stage concept ensures secure transmission of data from the vehicle(s) to the VSOC (cf. Figure 9.7). Firstly, a private communication channel between the vehicle and any backend servers based on VPNs is essential. For example, OTA updates and remote services (e.g., map data) require a private connection to the backend. In contrast to accessing VPNs via user account and password, as in typical IT environments, digital certificates (private and public keys) issued by a trusted certification authority protect the vehicle's access to the backend server infrastructure. The vehicles possess the public key of the backend and an individual private and public key. At the start of the connection, the backend and the vehicle exchange their certificates and check their validity. After successful verification, a private (encrypted), transparent tunnel is established.

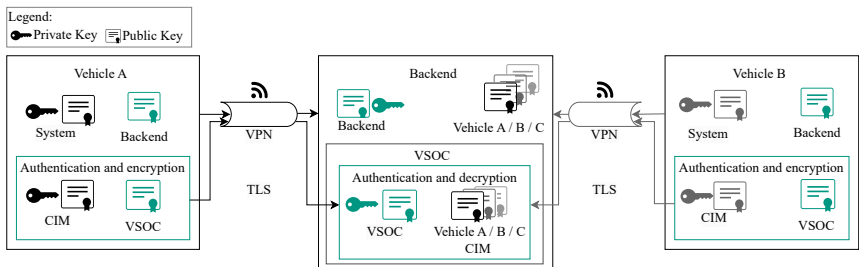


Figure 9.7: Secure channel for transmission of cybersecurity monitoring data to the VSOC.

A second layer of security assures that compromised components in a vehicle cannot easily manipulate the monitoring data sent to the VSOC. A TLS connection (via TCP) with two-way authentication ensures that as long as the CIM component is not compromised, no other component can send manipulated monitoring information directly to the VSOC. During initialization of the secure TLS channel, the vehicle verifies the certificate of the VSOC, and the VSOC verifies

the vehicle's certificate. Afterward, only the VSOC can decrypt the encrypted monitoring data. As a result, the vehicles require an additional private and public key for the respective information management, stored separately from the VPN keys.

9.4 Connection with additional information in the VSOC ⑤

After secure transmission (cf. Section 9.3.2), the recorded data is stored centrally in the VSOC's KG database, where additional information, including CTI and development artifacts, enriches the fleet's data. The database uses an LPG formalization¹⁰, as it offers greater schema flexibility compared to relational databases or time series databases and can be scaled well to process the growing data volumes efficiently (cf. Section and Section 9.4.1). This comprehensive database constitutes the foundation for the subsequent triage and analysis processes of the operations process (cf. Chapter 7), in which security-relevant events are efficiently evaluated and investigated.

Before being stored, preprocessing documents and the vehicles' incoming data extracts the information and relations (cf. Section 9.4.2) corresponding to the KG's data model. The data model is the essence of enriching the fleet's data with other information (cf. Section 9.4.3). It describes all connections between the considered information, defining the semantics of all queries of users within the VSOC.

¹⁰ cf. page 109

9.4.1 Labeled Property Graphs as a Knowledge Graph of cybersecurity information

Proposing a graph-based storage and query schema instead of established relational databases has several advantages (cf. Section *Database technologies* on page 108). The main reason for this choice is the schema flexibility and the focus on relationships between entities. The specific advantages of the LPG realization arise in particular from the lean modeling with *labels* and *properties*. Formally, an LPG is a directed graph $G = (V, E)$ where V is a finite set of *nodes* (or vertices) and $E \subseteq V \times V$ is a finite set of directed *edges* between nodes. Each edge $e \in E$ is an ordered pair of nodes (v_i, v_j) where v_i is the source and v_j is the target. Each node $v \in V$ has a set of *labels* $\mathcal{L}(v)$ and a set of *properties* $\mathcal{P}(v)$, where each property is a key-value pair. Similarly, each edge $e \in E$ has a *label* $l(e)$ and a set of *properties* $\mathcal{Q}(e)$, where each property is also a key-value pair.

As a result of this formal model, relationships and nodes in LPGs databases have searchable names (labels $\mathcal{L}(v)$ and l_e), which is an intuitive semantic that is adaptable to match the thought concepts of the VSOC personnel. Also, inhomogeneous properties are possible, allowing flexible and resource-saving storage by only storing the properties relevant to particular graph entities without additional tables and unnecessary empty fields.

Although, in general, graph databases are flexible, LPGs in particular support polymorphism by modeling multi-type entities. Polymorphous entities enable queries from different perspectives or abstraction levels, which is beneficial. While polymorphism is a common pattern in object-oriented programming to access certain properties depending on the application context, LPGs implement polymorphism with several labels $\mathcal{L}(v) = \{l_1, l_2, \dots, l_v\}$ for a node v . For example, while all logical components v_c have the label $\mathcal{L}(v_c) = \{l_c\}$ with $l_c = \text{"Component"}$, some are software components v_s with $\mathcal{L}(v_s) = \{l_c, l_s\}$, $l_s = \text{"Software component"}$, and others are hardware components v_h with $\mathcal{L}(v_h) = \{l_c, l_h\}$, $l_h = \text{"Host"}$. When querying all nodes v where $l_c \in \mathcal{L}(v)$ ($l_c = \text{"Component"}$) both types of components (software and hardware) will be retrieved.

As the recently standardized GQL¹¹ is not yet widely established, the implementation of queries depends on the particular graph database. In addition, GQL and other languages support queries similar to SQL but may have different names. Therefore, the terms *Insert*, *Update*, *Delete*, and *Select* (cf. Figure 9.8), which are familiar from SQL, are used in the following descriptions of preprocessing and integration with the KG.

9.4.2 Information preprocessing and integration in the Knowledge Graph

In-vehicle information

Once authenticated with the vehicles, the VSOC receives and decrypts the monitored data. The subsequent storage process in the KG involves extracting semantic information as nodes and edges from the messages, as specified by the data model (cf. Section 9.4.3). For example, if an observable occurs on a specific host v_h in a vehicle, the system creates a node v_{o_l} to represent this observable and an edge $e = (v_{o_l}, v_h)$ connecting v_{o_l} to v_h . Both the newly created node v_{o_l} and the edge $e = (v_{o_l}, v_h)$ are classified by assigning them one or more labels $\mathcal{L}(v)$ and l_e respectively. These newly formed graph elements are added to the graph database via its query API (*Insert* operation). Notably, each incoming message from a vehicle necessitates to create (*Insert*) at least one new node and edge for persistent storage.

Development process and support organization

While the vehicles provide digital objects, the development and support processes supply different documents (cf. Figure 9.8), requiring adequate preprocessing.

¹¹ cf. page 111

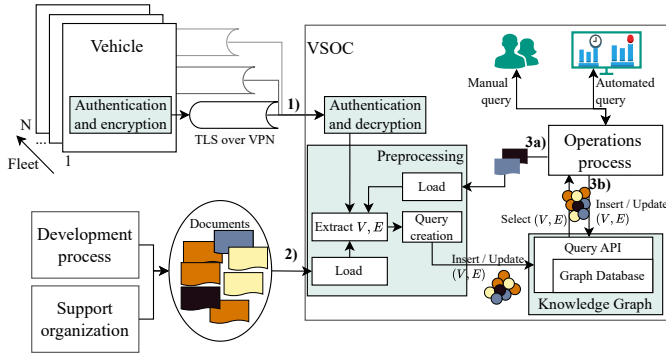


Figure 9.8: Integration of collected in-vehicle information and documents in the Knowledge Graph.

a) Development information Following a vehicle's production, the VSOC uses its development documentation to initialize (*Insert*) the corresponding information in the KG. Initializing a vehicle in the graph involves loading the files and extracting nodes and edges representing the vehicle's E/E architecture, logical architecture, variant management details, and associated risks. This initial population allows for the subsequent establishment of relationships between real-time in-vehicle monitoring data and the vehicle's design and configuration. For SDVs, where aspects of the E/E and logical architecture are dynamic, the vehicle provides this information during operation (cf. Section 9.1). In such cases, the database undergoes an *Update* to reflect these changes upon what was initially inserted.

b) Support organization The support organization provides reports which are initially isolated within the graph. When a new report document is loaded, the VSOC creates a corresponding node enriched with properties detailing the report's source. The triage process assesses the report's relevance, and if deemed necessary, establishes relationships, connecting it to affected vehicles, components, and other relevant nodes already present in the graph.

Operations process

The operations processes' interactions with the KG (cf. upper right-hand side in Figure 9.8) involve threat information, reports, and most importantly, incident and vulnerability analysis and handling information.

a) Documents Threat intelligence management and information sharing sub-processes prepare threat information and reports as documents. Accordingly, this information is loaded similarly to the development documents as soon as new knowledge is available. New information may be available in regular update cycles (e.g., updating threat information every six months) or at short notice (e.g., a particular threat actor launched attacks against a competitor).

b) Interactive Besides documents, the operations process interacts with the database via automated (e.g., repeated or rule-triggered) queries and manual queries via a command line interface. The VSOC staff formulates queries for the API of the database to display information (*Select*), analyzes it, and inserts new nodes or updates existing ones if necessary. For example, the triage phase of the operations process generates indicators, while incident handling updates these indicators to incidents. As each analysis may require different information, manual ad-hoc queries are necessary. Automated queries support the VSOC's tasks with repeatable interactions. Such queries may combine arbitrary operations (e.g., select, insert). For example, queries executed cyclically can update descriptive statistics or create indicators based on rules.

9.4.3 The ASCOT data model

The ASCOT data model [Gri+24c] presented in the following defines the links between the collected information to build a Knowledge Graph of automotive CSA. ASCOT stands for Automotive Security Context Toolkit - the data model provides the necessary context for security decisions in a VSOC. Majorly, the ASCOT data

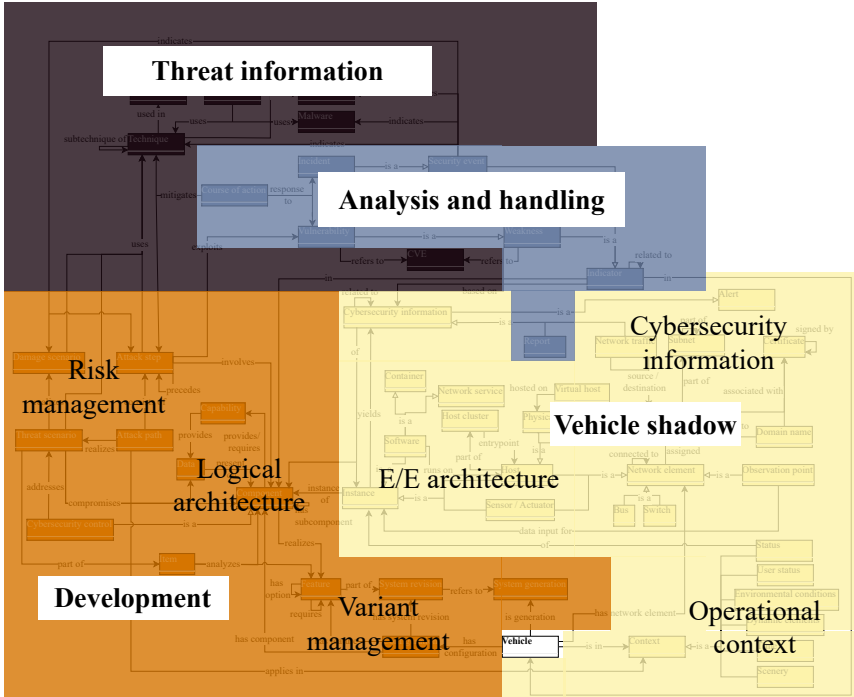


Figure 9.9: Overview of the ASCOT data model views (details cf. following figures)

model involves aspects of CRUSOE [Kom+18], feature-model based variant management (similar to the modeling of [Sch+23]), the operational context elements (cf. Section 9.2.2), the operations process (extending ISO/SAE 21434 [ISO21], cf. Chapter 7) and TTP based CTI modeling similar to MITRE ATT&CK®. Overall, the model connects the four proposed views of monitoring (cf. Section 9.1), where each view feeds a specific perspective of CSA.

An UML class diagram defines the views and relationships without detailing the classes' attributes. Because the model defines data, not an application, the CLASSES (cf. Table A.7) contain no methods. The colors of the classes symbolize the respective view (cf. Figure 9.9 and legend in Figure 9.10). Instances of the classes will be *nodes* $v \in V$ in the resulting graph G . The class **VEHICLE**

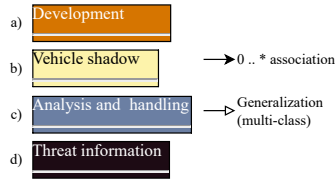


Figure 9.10: Color code (repeats colors defined in Table 5.2), connector legend and respective sections.

links different views and, as a central filter and search instance, is not assigned to any specific view. To model relationships, UML associations with unlimited cardinality and generalizations are used. Associations result in *edges* $e \in E$ of the graph. While the directed associations support reading the relationship, their navigability is not limited. Generalization indicates that an information (i.e., node v) is a member of several classes. If implemented as LPG, such nodes have several labels ($L > 1$), which allows to search for information from different viewpoints. Also, it accounts for escalation processes and unverified information. Initially marked as class A, information can also belong to class B after verification. The entire model can be found in the appendix (cf. Figure A.5).

a) Development The development view (cf. Figure 9.11) provides a unified perspective on the different vehicle variants, features and configurations, the components implementing features and their dependencies, and the known and assessed cybersecurity risks of the respective customer features. The proposed model derives from product-line engineering, SOA-based software architecture models, and TARA based on ISO/SAE 21434 and the proposed modified TARA methodology (cf. Chapter 8). Including the TARA outcomes in the KG is crucial for triage and analysis to identify risks and potential impacts. Moreover, relating incidents and vulnerabilities to the TARA enables an important use case, as continuous updates of risk assessments are required. The development view abstracts the E/E architecture which is part of the vehicle shadow.

Regarding the variants, the product-line approach highlights the differences and similarities of the vehicle variants with respect to customer features. While

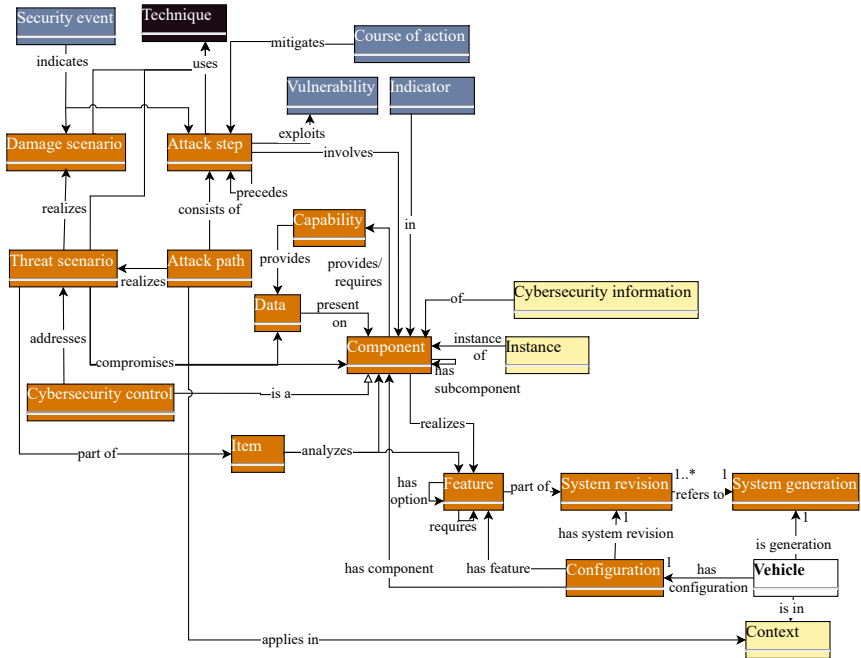


Figure 9.11: Development view.

variant management requires 150% models (all possible feature combinations), CSA focuses on 100% models—the configurations actually produced and chosen by customers. Reflecting the mission perspective of other CSA models, the vehicle's principal mission is the provision of customer features.

A VEHICLE has a particular CONFIGURATION, which has several FEATURES, resulting from selecting them in a particular revision, i.e., a version of the FEATURE at a particular point in time. Several VEHICLES may have the same CONFIGURATION. FEATURES are *part of* a corresponding SYSTEM REVISION, that changes over the life cycle of the product line. An update of the vehicle's software would reflect in a changed system revision of the configuration. However, a VEHICLE *has* exactly one SYSTEM GENERATION that does not change, e.g., a Golf 8 is a Golf 8 throughout its life, regardless of received software updates.

The relationship of FEATURES to each other corresponds to the configuration constraints of the product line. Feature models describe variability across a product line (cf. Section 2.2.3 on page 30). After production, however, the focus shifts to the commonality and inherent dependencies of a specific product instance (vehicle). Consequently, $\wedge$ (OR) and $\vee$ (XOR) constraints become less important; their function of constraining configuration choices is fulfilled after production. For example, a constraint like "feature A OR feature B" is relevant during selection, but once the product is built, whether A or B (or neither, if allowed) was chosen is simply a fact, not a lasting dependency. In contrast, $\wedge$ (AND) constraints, remain relevant as they describe the operational vehicle's functional dependencies. Retaining the $\wedge$ constraints, a particular FEATURE may *require* another FEATURE as part of a selected CONFIGURATION. Simplifying the original feature models, the relation *has option* (instead of detailed constraints) ensures that the development alternatives are still accessible in a tree-based manner. As a result, high-level conclusions may be drawn, such as affected features by a vulnerable COMPONENT, that *realizes* the feature. If incident response management intends to deactivate a feature, its dependencies must be considered.

Regardless of their implementation (hardware or software, cf. Section *Vehicle shadow* on page 198), COMPONENTS are modeled based on service orientation, and a component *has sub-components*. CAPABILITIES are *provided* or *required* by COMPONENT to realize features.¹² For example, *providing* a camera image is the CAPABILITY implemented with a SENSOR, while a SOFTWARE implementing object detection *requires* this capability. Capabilities provide DATA (data in motion).

Formalization as a capability is a data-centric description of a services' interface. However, modeling interfaces and data types separately from the service is also the standardized procedure in AAP, for example [AUT23c; Var19]. In SPLE, capabilities are part of the solution space.

¹² The capability-based modeling of services was developed in a joint publication [Sch+23]: M. Schindewolf, J. W. Wittler, T. Kühn, D. Grimm and E. Sax, "A Model-Based Approach to Automotive Feature Development for Updates and Upgrades," 2023 IEEE International Conference on Service-Oriented System Engineering (SOSE), Athens, Greece, 2023.

Definition 9.1: Capability

A capability represents what a component can do. It defines a specific interface that the component offers or consumes, described in terms of the data it handles and the mode of interaction (e.g., publish/subscribe, request/response).

Besides being used in a capability, data can also be *present on* a component as a configuration or database (data at rest). Components have a version as a property, because finding publicly known vulnerabilities in software used in a company's own products requires version information. From a risk management perspective, components and data are the assets of an item, and a cybersecurity property such as integrity, availability, or non-repudiation may be *compromised* by a THREAT SCENARIO. Components can be a CYBERSECURITY CONTROL.

The first step of TARA is to define the scope as an ITEM definition. The item *analyzes* a FEATURE or one or several COMPONENTS. One or more ATTACK PATHS with a particular feasibility rating as property realize a THREAT SCENARIO. A threat scenario contains a risk value as an attribute, and *compromises* a cybersecurity property of an asset (i.e., data or component). An ATTACK STEP *precedes* another attack step and *involves* a COMPONENT. Attack paths, *consisting of* several attack steps therefore model multi-step attacks. A threat scenario realizes one or more DAMAGE SCENARIOS. Each damage scenario has an impact rating attribute. The ASCOT data model extends ISO/SAE 21434 by the required environmental conditions for attacks by modeling that an attack path *applies in* a certain CONTEXT. For example, attacks via the OBD interface require connectivity to an external diagnostic device. Besides, the model considers that the extended TARA methodology classifies each damage scenario, attack step, and threat scenario as a technique from the threat catalog. In the ASCOT data model, an attack step *uses* a TECHNIQUE.



However, all observables (cf. Table A.5) from the various sensors (cf. Figure 9.2) can be used as derived classes. Hence, the analysts can introduce new observables or modify existing ones without adaptation of the data model.

The model developed to mirror the vehicle's network can represent both traditional E/E architectures with legacy buses and SDVs. It reflects physical interconnectivity and software-hardware dependencies (cf. Figure 9.12 on page 198), covering hardware and software architecture, the network topology, and showing which software runs on which host. For orchestrated software execution or cloud service connectivity, the in-vehicle information dynamically updates this view.

The physical context of the vehicles represented in the KG follows the proposed elements of the operational context (cf. Figure 9.4) based on ISO 34503. Although the mobility of IT hosts increases (e.g., remote work, smartphones), a vehicle's mobility is the key operational factor, which requires being represented in a vehicle shadow. The operational context enhances the understanding of observables, as it allows to assess physical impacts and potential causes of incidents, and helps to identify false alarms from IDS due to unusual vehicle behavior, which might not be evident from network data alone.

The ASCOT data model treats microcontrollers (real-time OS or no OS) and microprocessor-based execution platforms (with POSIX or other high-level OS) in the network equally as HOSTS that execute software. Hosts can be either physical or virtualized, while VIRTUAL HOSTS are *hosted on* a PHYSICAL HOST. If the E/E architecture contains an orchestrator, several hosts form a HOST CLUSTER with dynamic software allocation. Usually, such a cluster has a main host (*entrypoint* for cluster) that accepts control commands and other hosts that are *part of* the cluster as workers. In addition to hosts, other hardware exists in vehicles, namely SENSORS AND ACTUATORS. A further element of the E/E architecture is SOFTWARE (including CONTAINERS), that *runs on* a host. Still, traditional ECUs can be modeled, for example, modeling microcontroller OS such as AUTOSAR classic [AUT03] and the application code as separate software nodes, and the microcontroller as a HOST. Hosts, software, and sensors/actuators are all described

as INSTANCES of a COMPONENT to link the development view with the technical implementation that is scaled to many vehicles.

The physical network topology is formed by NETWORK ELEMENTS. Each hardware element (sensor/actuator, host) is a network element, *connected to* other network elements, yielding a flexible way to describe the whole architecture. BUS systems (e.g., CAN) are modeled as separate nodes, highlighting that all connected elements can communicate with each other via the bus. Further network elements such as Ethernet SWITCHES, routers, and other wireless communication interfaces, which do not have a computing or sensing/actuation capability, complete the physical networks.

Network configuration consists of NETWORK ADDRESSES, that model IP and MAC addresses or port numbers, assignment of IP addresses to SUBNETS and DOMAIN NAMES, and assignment of IP addresses and domain names to CERTIFICATES. This information describes known, permitted communication endpoints (especially within the vehicle) and potentially unknown endpoints, including certificates used for encryption or authentication. NETWORK TRAFFIC usually has a *source* and/or *destination* NETWORK ADDRESS, which may be an IP address in Ethernet networks but may also indicate CAN identifiers.

The operational context centers around several subclasses derived from the CONTEXT base class. SCENERY, DYNAMIC ELEMENTS and ENVIRONMENTAL CONDITIONS form the ODD or COD, while the ODC comprises the ODD along with the STATUS and USER STATUS classes. The CONTEXT also connects the design time (assumptions in TARA) with the vehicle's current operational conditions and domain. The STATUS *of* an INSTANCE, for example, is a context. Status includes, the activation status of a service, shut-down hardware, or an error condition, including DTCs. TIME completes the considered aspects of the operational context. The elements of the operational context require refinements as soon as more experience is available regarding which CONTEXT information is most helpful for CSA. This structure can also be extended with additional subclasses if required.

CYBERSECURITY INFORMATION is mainly generated by cybersecurity monitoring sensors within the vehicle. The vehicle shadow view therefore accumulates all

observables collected from the vehicles. Although the class CYBERSECURITY CONTROL (cf. Section *Development* on page 194) models monitoring sensors and preventive cybersecurity measures, the INSTANCES of the respective components are the implemented measures that *yield* the information. Some tools, e.g., IDS and firewalls, gather input from NETWORK ELEMENTS functioning as OBSERVATION POINTS. Via the relationship cybersecurity information *of* COMPONENT / INSTANCE, each observable or report can be assigned to one or more elements of the E/E architecture. This relationship describes the generation of the information, such as application logs, and can represent access and provision (e.g., for resource usage or file access). The triage process creates INDICATORS, marking security-relevant information. An intermediate step can be the generation of ALERTS, which indicate algorithmic preprocessing of CSI, such as by an IDS, or with SIEM rules.

c) Analysis and handling The incident and vulnerability analysis and handling view (cf. Figure 9.13) connects current and historical cybersecurity issues. Generalization relations are used to associate the terms of incident and vulnerability handling, according to the (re-)definitions of the cybersecurity operations process (cf. Section 7.1). The aim is to support incident responders and vulnerability assessors within this process. An INDICATOR is the result of the triage that initially determines the relevance of cybersecurity information. Cybersecurity information primarily includes observables from the vehicle fleet (cf. vehicle shadow). A further source may be REPORTS from manufacturer-internal or external sources, such as vulnerability scanning or bug bounty programs. The REPORT class allows the inclusion of textual, unstructured information on the same level as observables generated within a vehicle. An indicator is either a SECURITY EVENT or a WEAKNESS. Security events and INCIDENTS can also *refer to* weaknesses and VULNERABILITIES if an exploitation is involved. Indicators are *in* a COMPONENT (particularly vulnerabilities) or in a VEHICLE. The connection to the VEHICLE enables retrieval of related vehicles and all incidents and vulnerabilities within a given vehicle with only a few query statements. COURSES OF ACTION are a *response to* incidents and vulnerabilities. Storing responses in a database helps

to create playbooks¹³ and track the effectiveness and efficiency of the response process. This work proposes to transmit the course of action over the air to the vehicles, as described in detail in the following chapter (cf. Chapter 10).

Security events can *indicate* different things: TOOL, TECHNIQUE, MALWARE, DAMAGE SCENARIO or ATTACK STEP if signs of the usage, impact, or attacker behavior are present. Aligning with the attack steps and damage scenario connects this view to risk management within the development view. The alignment enables the identification of required TARA updates (cf. Section 7.3.1) and using precalculated risk assessments within TARA for the prioritization of responses. Weaknesses and vulnerabilities may *refer to* CVE entries.

d) Threat information The final view focuses on the strategic perspective (cf. Figure 9.14) by integrating CTI. It also acts as an incubator for gathering internal knowledge and serves as a basis for continuously improving threat information

¹³ a repeatable sequence of actions depending on the type of incident

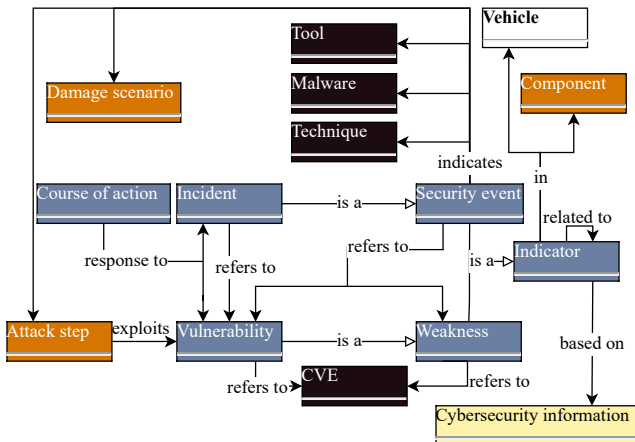


Figure 9.13: Analysis and handling view.

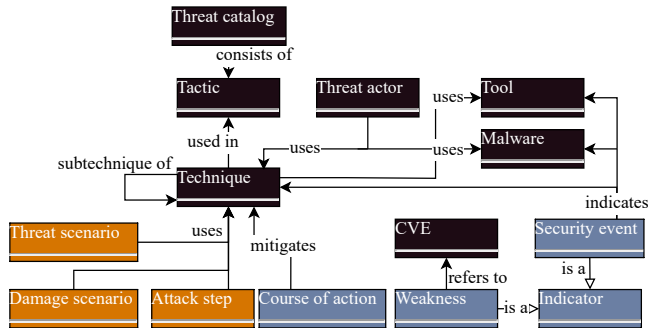


Figure 9.14: Threat information view.

(cf. Section 7.3.4). Here, INCIDENTS and VULNERABILITIES are connected to CTI data based on modeling TTP.

As such, this view centers around the **TECHNIQUES** that are *used in* one or more **TACTICS**, and the **THREAT ACTORS** that *use* the techniques. To increase the level of detail, *sub-techniques of* a technique can be described. Incidents that *indicate* a technique are the procedures in the TTP formalism. Several **THREAT CATALOGS**, consisting of tactics and techniques, may be relevant to a particular OEM. For instance, catalogs may distinguish distinct versions or TTP sets targeting specific types of vehicles (e.g., passenger cars and trucks). **TOOL** and **MALWARE** differentiate the (potentially malicious) use of legitimate tools, such as command-line tools built into POSIX-based OS, from the execution of software known to be malicious.

9.4.4 Potentials for the Use of Artificial Intelligence

Cybersecurity operations in automotive environments demand timely detection, sound analysis, and effective response under conditions where data is heterogeneous, attacks are evolving, and labeled examples are scarce. The KG proposed in this work provides a structured foundation to address these challenges. Filled with the data an OEM is expected to have at hand from development and fleet operation, it provides the right substrate for modern AI techniques, opening up a

range of powerful applications [Eck+25; Sik23; Ma+24]. The following outlines these opportunities, spanning both the construction of the graph and its application in cybersecurity operations, structured along the processes of *triage*, *incident analysis*, and *incident management* (cf. Section 7.2) [Gri+23b].

Construction and Integration

Extracting information from unstructured sources is one of the core capabilities of modern AI components such as LLM¹⁴. In the context of KG construction, the relevant tasks are *named entity recognition* and *relationship extraction*. Extracting CTI entities and their relationships from unstructured text has been demonstrated using both cybersecurity-adapted [MS25] and general-purpose LLMs [Zha+25], the latter without requiring labeled training data. In industrial environments, similar approaches have been used to fuse heterogeneous security data sources into a unified KG for asset risk identification and attack path analysis [Nan+26]. Beyond threat descriptions, LLMs are also used to parse system logs as input to a KG, enabling natural language querying over heterogeneous security data [KKE24].

The scarcity of labeled attack data poses a fundamental challenge for learning-based approaches in automotive contexts. Human-in-the-loop learning (preference-based active learning), where analysts provide preference feedback by selecting among system-generated candidates to iteratively refine KG-based AI models such as to classify or rank nodes, offers a promising direction to address this. This paradigm, introducing minimal labeling effort to the users, was established in RL¹⁵ [Chr+17] and recently popularized through its application to LLMs [Ouy+22]. In general, suitable processes to develop, select, and integrate AI components as well as keeping them functional are necessary [Sto+20b; Pis+20; Zin+26], which might require further research targeted to cybersecurity operations as well.

¹⁴ Large Language Models (cf. glossary, page 327)

¹⁵ Reinforcement Learning

Cybersecurity Operations

a) Triage In the triage phase, AI-based support of analysts includes prioritization or ranking of alerts and information, and detection methods (similar to IDS). For example, a rule- and pattern-based approach builds a KG from audit logs, enabling both detection of known attack patterns and reconstruction of attack graphs [Kur+22]. Complementarily, a ranking-based approach measures the degree of maliciousness of entities in a SIEM-based KG to prioritize indicators for further handling [Naj+19]. Moving toward learned representations, graph embedding techniques [CZC18] enable GNN¹⁶ to classify nodes based on their relational context rather than isolated features, applied both to network traffic [Lo+22] and to interactions between system entities, where anomalous behavior is identified without requiring prior attack signatures [Zen+22]. An embedding-based approach was also applied in an industrial systems application [GDF21].

b) Incident Analysis Beyond triaging individual events, incident analysis requires abstracting low-level observations into meaningful behavioral patterns and assessing their significance. KG embeddings have been used to infer the contextual semantics of audit log events, enabling abstraction of low-level observations into clustered high-level behavior signatures that generalize across attack variants [Zen+21]. Reasoning over a threat intelligence KG enables the generation and ranking of attack hypotheses: given partial observations, probable unobserved attacker techniques are inferred from the graph structure [Kai+23]. Extending toward prediction of future attacker steps, LLMs combined with KG traversal have been used to automatically generate multi-step attack scenarios for red team testing [Loe+25].

¹⁶ Graph Neural Networks

A cross-cutting opportunity lies in making KG-based reasoning accessible to human analysts. LLMs can translate natural language queries into the KG's native query language (e.g., Cypher), enabling analysts to retrieve and explain system states without requiring graph query expertise. Combined with KG embeddings and vector indices for retrieval-augmented generation¹⁷, this creates an explainable interface layer over the operational knowledge encoded in the graph [KKE24], an aspect that purely statistical AI approaches cannot easily provide. However, research into methods for building trust in AI systems is also advancing, including verification [Gri+24a] and the rejection of predictions in cases of uncertainty [ZGS26].

c) Incident Management Among the tasks of incident management is to select and implement suitable responses to confirmed incidents. This problem can be treated as a decision support or recommendation problem with the KG as information basis. The CRUSOE toolset [Hus+22] represents one of the most comprehensive operational implementations of KG-based decision support to date, applying a Bayesian network over a graph-based system model to evaluate configuration resilience and recommend hardening measures. RL-based agents have also been applied to select containment and defense actions over graph-structured network representations, with attention mechanisms further improving adaptability to topological variation [KBH25; San+25]. However, current approaches remain limited to simulated environments and synthetic attack scenarios. Closing the gap to real traces of malicious activity on heterogeneous graphs remains an open challenge. Nevertheless, the automotive industry has several years of experience in researching RL agents, particularly with regard to approaches for fully automated driving [Sta+20], which might be helpful to explore this new use case.

¹⁷ Instead of relying solely on generated text, LLMs can improve factual accuracy to natural language questions by retrieving semantically relevant nodes from a graph database using their embedding representations.

10 Immediate response over the air

Currently, ISO/SAE 21434 considers software updates (code change) or configuration updates (software parameter change) to respond to vulnerabilities or incidents, but updates of safety-critical functions require a new approval. Even if the testing phase is automated, the entire process of development, testing and homologation takes a considerable amount of time. Fast and iterative reactions to critical vulnerabilities or ongoing attacks are therefore impossible. For immediate responses, any changes requiring homologation have to be avoided.

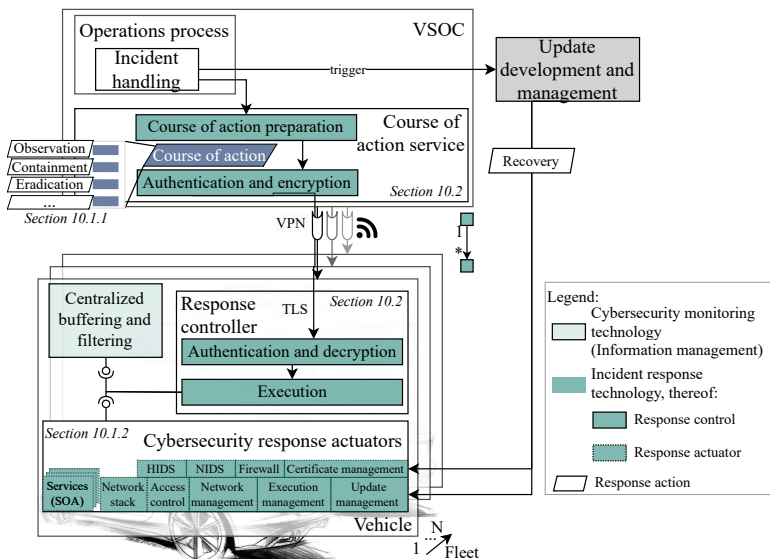


Figure 10.1: Overview of the immediate response approach.

Therefore, this work proposes to pre-homologate responses, which are activated over-the-air and executed by security actuators (cf. Figure 10.1) immediately (cf. ⑥ in Figure 3.1 on page 57). Multi-stage reactions can be built up as a course of action consisting of several steps. The response control component in the vehicles parses the course of action commanded by the VSOC, to trigger the desired changes of the corresponding actuators. Activation and execution of responses can be specified, tested and approved during development. This implies having to consider responses during development to enable immediate actions when it is necessary.

10.1 Pre-homologated responses

10.1.1 Analysis and categorization of response actions

To identify suitable response actions and actuators, the measures established in IT were examined (cf. Table 4.2 on page 96). Overall, some IT measures cannot be implemented for the approach proposed here, as they are not suitable for development-time homologation, or not applicable due to constraints of a safety-critical embedded system. For example, deliberately redirecting an attacker into a sandbox to monitor its activities is presumably not practicable, as the corresponding resources are lacking and the system must remain continuously available. Other examples include the targeted removal of malware by selective deletion from memory, or the recovery of particular compromised files. Such actions can hardly be homologated in advance, as they would require targeted write operations. Other measures cannot be implemented as in IT, but there are possible alternatives from automotive research and from the given approaches for the development of SDV to achieve the same goal. Instead of completely shutting down the system, for example, the limp mode can be activated to achieve containment of the possible effects.

Resulting from the analysis, a categorization of possible response actions was developed (cf. Figure 10.2), which classifies all suitable responses according to the

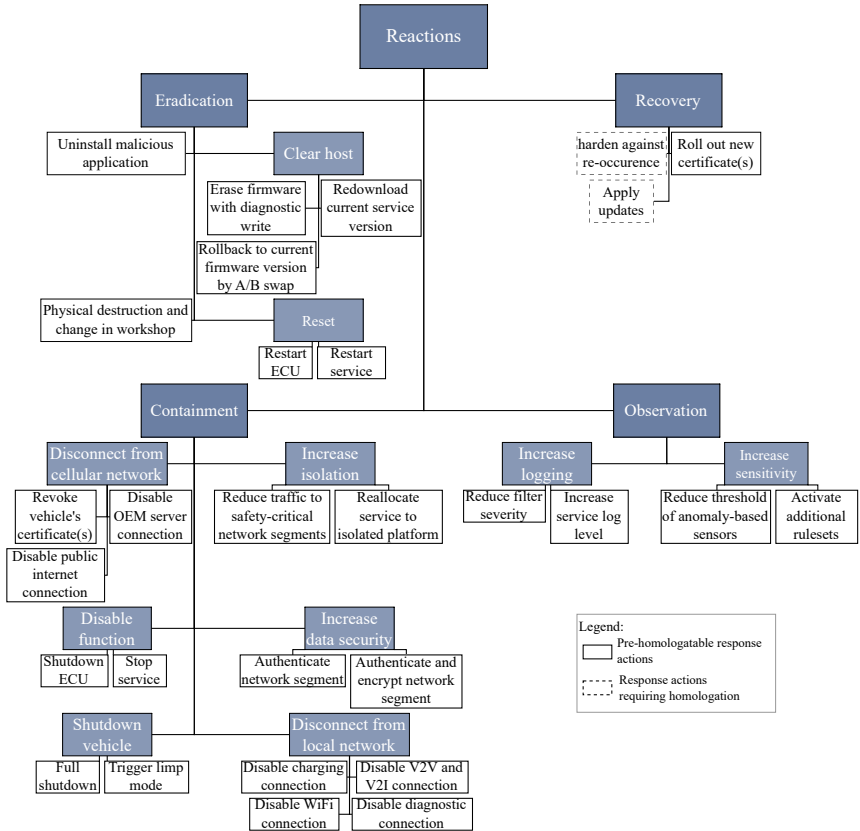


Figure 10.2: Categorization of response actions.

objectives of containment, eradication, recovery, and observation. The category of observation actions was added with the aim of increasing the observability of incidents.

10.1.2 Response actuators

The identified responses can be implemented in the vehicle using several actuators. A cybersecurity response actuator is a component that performs responses by switching between different, pre-implemented configurations during operation. Note that response actuators should not be confused with physical actuators (cf. vehicle shadow view, Section 9.4.3) like motors. Essentially, the elements of the vehicle used for monitoring are suitable as cybersecurity response actuators. The appendix contains a detailed assignment of considerable actuators to respective actions (cf. Table A.8).

Containment is the essential task of (incident) response, because it can increase the available time to develop an update, before critical impacts may occur. Firewalls are of particular importance, as they can be used to implement the majority of containment actions. For example, pre-configured rules to allow or block connections to particular servers, such as the OEM's backend, the public internet, or the VSOC, can be implemented. In addition, the SDV technologies of execution management can introduce more flexibility to the response capability.

Observation actions help to implement an aspect of the incident response plan, a prepared procedure for the loop of incident handling to monitoring. Therefore, observation actions accelerate the access to additional information, when required. This thesis proposes a centralized buffering and filtering mechanism with a configurable log level (cf. Section 9.3.1), which implements an interface for observation actions.

Eradication is a challenging task to pre-homologate, as it involves deletion of information, which may have critical consequences as well. Classic ECUs offer an advantage here, as usually, they do not have a file system. Restarting the ECU therefore terminates active exploitations, unless the program memory has already been updated for persistence. Besides, the automotive-specific update management services could be used for eradication, e.g. with UDS or via an A/B swap to a previous software version.

Recovery actions include the responses that require new approval (shown with dashed lines in Figure 10.2). In addition, new certificates can be rolled out, which can be done during operation without new homologation. Applying the developed updates and hardening by adapting cybersecurity measures are the final recovery actions to restore full functionality, which is performed by the Update Management.

10.2 Response activation over-the-air

To make use of pre-homologated responses, they are activated over-the-air from the VSOC. In the overall workflow (cf. Figure 10.3), the response control component is responsible for executing reactions in the vehicle. Before being executed, however, the operations process must determine the required reactions, namely in the incident handling phase. Hence, before the technical details of the response control (cf. Section 10.2.4), such as its interface, are discussed, the integration into the operations process is clarified (cf. 1) in Figure 10.3). To realize the responses determined from human analysis, they are prepared digitally as a course of action using a course of action service component in the VSOC (cf. 2) in Figure 10.3). This specific component also establishes the secure connection (cf. Section 10.2.3) to the response controller in the vehicle.

Setting the course of action via the secure transmission channel (cf. 3) in Figure 10.3) makes use of the standardized interface (cf. 4) in Figure 10.3, detailed in Section 10.2.4) provided by the response control, to which the VSOC attaches. After the course of action service transmitted the course of action over-the-air, the response controller component executes (cf. 5) in Figure 10.3, detailed in Section 10.2.5) the responses through triggering the respective response actuators.

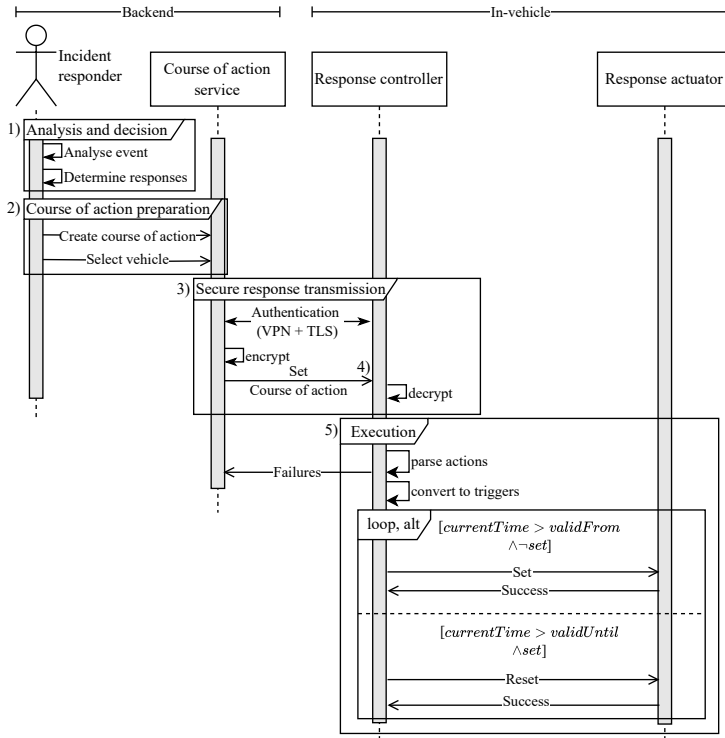


Figure 10.3: Workflow to activate pre-homologated responses over-the-air using response controller and course of action service components

10.2.1 Analysis and decision

The incident handling phase becomes active when, based on the data collected with holistic monitoring, a security-relevant event was identified in the triage phase. When analyzing the event (cf. 1) in Figure 10.3), the incident handler focuses on quantifying the extent of the damage and identifying the attack vector. If the incident analysis determines that the event is indeed an incident requiring a response, incident management becomes necessary (cf. Section 7.2.5).

The aim of the incident management process is to develop and implement an incident response plan (cf. Section 7.2.1 and Section 7.2.5). In addition to defined responsibilities and a communication strategy, the central aspect of the incident response plan is defining the remedial actions (cf. a) in Remark 2.5). Depending on the damage and attack vector, different measures are considered when planning the response, for example interrupting the connection to OEM servers if it is an attack via backend systems. Based on the defined response categories *containment*, *eradication*, *recovery* and *observation* and the pre-homologated response actions, possible options are evaluated. As a result, a set of responses is determined (cf. 1) Figure 10.3) that shall ensure a precise and targeted response to the incident, minimizing damage and quickly restore the functionality of the affected system. In the approach proposed here, the determined responses resulting from incident handling can be composed of the pre-homologated actions. Future work should further investigate how the selection process can be supported. For example, virtual tests could verify the effectiveness of the selected responses.

10.2.2 Course of action preparation

After these human analysis and decision-making tasks (possibly supported by software), the technical realization of the selected responses proceeds. On the VSOC side, a course of action service (tool, software) is a necessary technical part to connect to the vehicles' response controller. The course of action service provides a user interface for the incident responder (cf. VSOC roles, Section 4.1.1), e.g. graphically or by reading in a file to generate the desired responses as a course of action (cf. 2) in Figure 10.3). It also enables the selection of particular vehicles as the target for the course of action. In this work, a prototype graphical interface to select certain response actions was implemented (cf. Section 11.5.2).

10.2.3 Secure response transmission

The course of action service establishes a secure connection to the response controller component of the selected vehicles using authentication and encryption

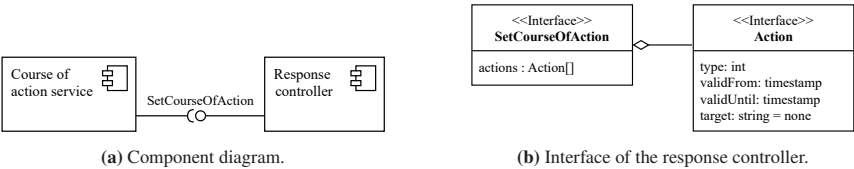


Figure 10.4: Interaction of response controller and course of action service.

to transmit the course of action. The interface for setting the course of action (cf. 4) in Figure 10.3) and its execution (cf. 5) in Figure 10.3) are discussed in the following sections.

To transmit the reactions to the vehicles during operation, an over-the-air communication channel between the vehicles and the VSOC is required, for example a WiFi or cellular connection. The VPN-based connection previously used to collect monitoring data is the prerequisite to secure the transmission, as it limits the amount of hosts that are able to send responses to the vehicles to the hosts with access to the VSOC network. A TLS connection between the course of action service (VSOC) and the response controller (vehicle) component is established via the VPN channel. For an illustration, please refer to Figure 9.7, as the structure for the secure transmission of responses is analogous. As for monitoring, a specific certificate is required for each vehicle and one for the course of action service, in addition to the certificates for the VPN connection. These are used for mutual authentication and subsequent encryption when the connection is established.

10.2.4 Setting the course of action

The response control component has a single interface `SetCourseOfAction`, which is accessed by the VSOC's course of action service component (cf. Figure 10.4a). A course of action consists of a list of actions (cf. Figure 10.4b), each of which has a `type`, a validity period for activation (`validFrom` and `validUntil`), and an optional `target`. The `type` identifies one of the implemented pre-homologated responses. The activation period, described by start and end times, enables persistence of responses instead of retransmitting

the currently required actions from the VSOC to the vehicles after each restart. Besides, defining the end of reaction validity enables to re-establish the connectivity to the backend after a defined time if a response completely locks down the vehicle from external communication. The optional parameter of the interface is required for certain actions that can be applied to different targets. For example, the actions *Reallocate service to isolated platform* and *Shutdown ECU* can be applied to different services or ECUs, which is why the action must be specified more precisely using the target parameter.

10.2.5 Execution

The execution sequence (cf. 5) in Figure 10.3) starts after setting and decrypting a new course of action. First, the type, timing and possibly the target are analyzed for each action in the list. Note that the response control abstracts the implementation of particular responses via the action types.

First, while the abstraction enables uniform, coordinated responses across different vehicle models and generations, some response types may not be supported in all vehicles, in particular due to cost-benefit considerations or because they are not relevant due to non-existent interfaces. After parsing the types, unsupported or incorrectly entered actions are thus reported back to the course of action service component via the secure channel (cf. failures in Figure 10.3).

Second, due to the different E/E architectures, including the available response actuators, it depends on the vehicle variant with which configuration change a certain action can be realized. For example, one model may connect to cellular networks via a telematic control unit, while another may connect via the infotainment system. Accordingly, *disable public internet connection* would require different configuration settings in the two vehicle models. Hence, all executable (supported) response actions are converted to two triggers (set, reset), sorted according to their timestamp. In this context, a trigger is a plan to set or reset a specific configuration of a particular actuator, where the validity period of the response action determines the time of set and reset.

As soon as a `validFrom` timestamp of an action is reached, the desired configuration setting of the actuator is activated via a SOA interaction (e.g. service call) and feedback is sent to the response control component if the action was successful or not. Successfully triggered actions are reset when their `validUntil` timestamp is reached.

11 Prototypical implementation of monitoring and response approaches

While the presented approaches cover both processes and technologies, this section concentrates on the prototypical implementation of selected aspects. The core of this chapter is to showcase holistic monitoring and the procedure for defining monitoring requirements for a specific item, the adaptive headlight system. This system was already an example in ISO/SAE 21434. Another example focusing on illustrating the procedure for specifying monitoring requirements is discussed in the appendix (cf. Appendix B).

The adaptive headlight system was implemented in simulation and a miniature vehicle (remote controlled truck in scale 1:14) using the SOA middleware ROS 2. All software modules presented in this chapter were implemented as containerized ROS 2 services, compiled via the GitLab CI/CD system, and built as container images. The images are downloaded and started on the target systems, so that no software needs to be installed apart from the container runtime.

In addition to vehicle capabilities such as the adaptive headlight system, software is also required for the considered cybersecurity operations activities, i.e., monitoring and response. Particularly, the implemented system includes centralized buffering and filtering, operational context monitoring, storing the data in a KG (cf. Section 11.4.3), and response actions for containment and observation. The resulting software contains an authentic vulnerability to realize an attack path similar to the ones pinpointed in ISO/SAE 21434. Incident analysis serves as an exemplary use case of the KG.

11.1 Example item: Adaptive headlight system

Remark 11.1: Adaptive headlight system

An adaptive headlight system has several tasks following the description in ISO/SAE 21434 (Annex H). First, it turns the headlight on or off by driver demand via a physical switch or button. If the headlight is in high-beam, the system switches the headlight automatically to low-beam when an oncoming vehicle is detected. It returns the headlight to high-beam if the oncoming vehicle is no longer detected (derived from [ISO21]).

A DFD of the implemented headlight system describes the logical architecture of the item (cf. Figure 11.1). The DFD comprises a segment of the entire logical architecture of a vehicle, defining the boundary of the item (cf. TARA terminology, Section 2.3.1). In addition, it shows the result of the asset identification. All processes (circles) and data flows (arrows) within the item boundary represent assets. As an assumption, some primitive cybersecurity measures are implemented, including a filtering functionality in the gateway, allowing only particular messages of valid senders to pass to the bus. Also, the TCU¹ has no direct access to the vehicle's network, and external networks connect via a firewall (deployed on the TCU, cf. technical integration, Section 11.2.1) to the in-vehicle network. For example, the dashboard host is allowed to connect to the public internet via the firewall and TCU.

One of the item's assets is the headlight power actuator, which activates the headlamps. The camera recognizes oncoming cars and the illumination. The information is forwarded to the headlight controller, together with the status of the headlight toggle button. Depending on illumination, oncoming vehicles, and the driver's demand, the headlight controller decides the target status of the headlight. It sends requests to the headlight power actuator to change the status (high-beam, low-beam, off) via the gateway.

¹ Telematic Control Unit

11.2 Technical platforms and vehicle variants

To demonstrate the advantages of the KG approach for a VSOC, recording the data of a single vehicle is not sufficient. Accordingly, a "fleet" of three vehicles was set up, consisting of (A) the miniature vehicle (cf. Section 11.2.1) and two vehicles with simulated E/E architecture (B/C). The three vehicles also realize different variants (cf. Section 11.2.3). Parallel operation of the three vehicles aims to illustrate the fleet-wide monitoring approach by inserting the information in parallel into the KG. The operation of the three vehicles is synchronized within an environment simulation using CARLA [Dos+17] (cf. Section 11.2.2). As a result, the functions of all vehicles can make use of simulated sensor data.

11.2.1 Miniature vehicle

The miniature vehicle (A) is a 1:14 scale Mercedes-Benz Arocs dump truck. The E/E architecture that was added to the electric drive system features two NVIDIA Jetson AGX Orin (named *center* and *wing*), which are installed in a modular box

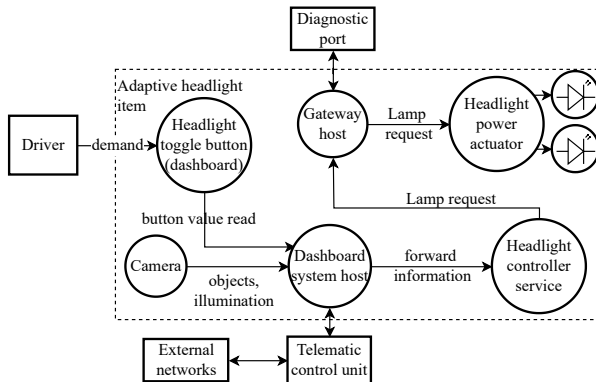


Figure 11.1: Data flow diagram of the adaptive headlight system item, derived from ISO/SAE 21434 and adapted to actually implemented system.

on the loading area, powered via a PoE²-enabled switch. The vehicle's built-in electric motor and LED lighting were connected to one of the NVIDIA hosts via an I2C³ bus. The NVIDIA hosts, which would be also available in a safety-certified version for industrial use, are equipped with a 12-core Arm® processor.

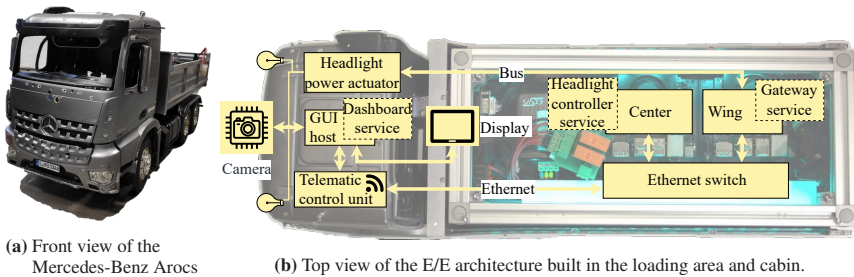


Figure 11.2: The miniature vehicle.

A Radxa X4 board with an Intel N100 CPU (four cores, x86-64 architecture) was installed in the driver's cab as the third physical host in the vehicle (named *tcu*), majorly providing infotainment functionality and connecting the vehicle to the wireless network. Connected via PoE to the modular box, the Radxa X4 runs two VMs using Proxmox virtualization environment (same software used to simulate the E/E architectures for vehicles B/C, cf. Section 11.2.2). In the miniature truck, it runs the firewall solution (cf. Section *Firewall and network management* on page 240), installed on the first VM (named *firewall*). A touch display is connected to the second VM (named *gui*) to display the dashboard application. Overall, the miniature truck has five hosts (three physical plus two virtual), leading to five nodes in the the KG.

For its driving task, a camera in the driver's cab focuses on the road ahead of the truck to enable lane detection. Being connected to the *gui* VM, the camera images

² Power over Ethernet (cf. glossary, page 328)

³ Inter-Integrated Circuit

are published in the ROS 2 middleware. However, in this dissertation the camera is not used for the implementation of the headlight assistant; instead, an object list gathered from the simulation is used. Besides the realized features, such as the adaptive headlight system, additional software and hardware is required for the physical control of the miniature vehicle, which is out of the scope of this chapter.

11.2.2 Simulated E/E architecture and fleet synchronization

Several tools and interconnections are required to simulate a vehicle fleet, which is discussed in further detail in a publication by the author of this dissertation [GSS23]. At the level of the E/E architecture, an individual vehicle is modeled as a virtual network of several VMs. The physical Ethernet network of a vehicle is simulated using Linux' network bridges, mimicking physical switches. Proxmox was used to create the virtual vehicle networks, which is a Debian-based open-source solution for managing VMs. The two simulated vehicles (cf. variants B/C, Section 11.2.3) comprise the same number of (computing) hosts as the miniature vehicle, leaving out the *tcu* physical host. Accordingly, four VMs are used per vehicle (*gui*, *center*, *wing*, *firewall*).

CARLA⁴ simulates the environment (sensors, actuators) of the fleet. Moreover, the three considered vehicles (plus additional background traffic) operate and interact in the same simulated world (cf. Figure 11.3). However, one major challenge is to synchronize the behavior of the vehicles, given their independent network and calculation. Usually, the CARLA simulator is controlled by a single client that dictates the simulation server when to calculate the next simulation step. As CARLA does not natively offer synchronization of multiple simulation clients, an additional synchronization server was developed, which waits for all clients' ready-signal and then triggers the next calculation step [GSS23].

⁴ version used: 0.9.15

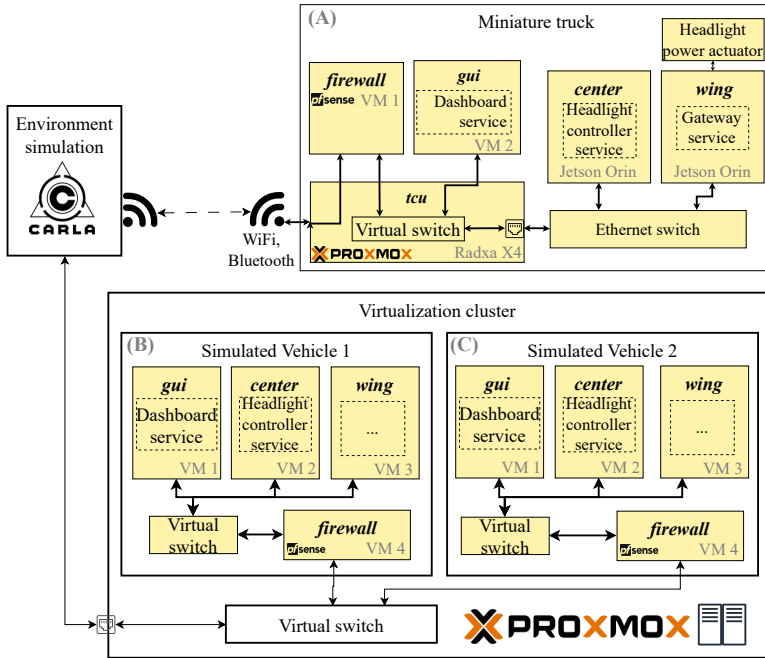


Figure 11.3: Simulated and miniature vehicle coupled within environment simulation.

Furthermore, to demonstrate the operational context monitoring, a simulation environment (simulation map) that replicates a real environment is required. In particular, the coordinates of the roads should correspond to the actual environment to enable correct assignment to an actual city, state, and road type. As such, the maps supplied with CARLA are not suitable. Accordingly, with funding from the ICM⁵, a simulation map was integrated into CARLA that represents the KIT campus and its surroundings (cf. Figure 11.4). To support simulating V2X features and attacks in the future, a V2X plugin for CARLA was developed, that is now part of the official open-source codebase [Gri+24b].

⁵ InnovationCampus Future Mobility (cf. glossary, page 326)



(a) Real KIT campus south.



(b) Simulated KIT campus south.

Figure 11.4: Comparison of real and simulated KIT campus.

11.2.3 Realized vehicle variants and capabilities

In addition to the considered item, the adaptive headlight system, other software components were developed to illustrate the technological contributions. Particularly, three different developed vehicle variants (A-C, cf. Figure 11.3) shall highlight the analysis of the causes and effects of the realized attack path. The variants differ in the following key aspects:

- A) Adaptive headlight system (vulnerable), sophisticated open-source automated driving stack
- B) Simple automated headlight system, simple waypoint-based automated driving
- C) Adaptive headlight system (vulnerable), simple waypoint-based automated driving

For each variant a dashboard application is available, that includes an instrument cluster and can be used to operate the lighting control and automated driving functions. Variant A was deployed on the miniature truck, while B and C were deployed to a simulated E/E architecture.

Headlight control

Two light control system options were implemented: the adaptive headlight system (HL-1) and a simple headlight system (HL-2). Both systems control the vehicle's lights (low beam and high beam) indirectly via a gateway and headlight power actuator to trigger the desired lighting conditions. The simulated vehicle variants (B/C) can only activate and deactivate the simulated headlights⁶, while in the miniature vehicle (variant A, cf. Section 11.2.1 on page 219), the physically installed lights are controlled.

The activation of the low beam depends on the brightness of the environment. In all three variants (A/B/C), respectively in both options (HL-1/L2), the brightness is sensed in the simulated environment to ensure comparable measurements for all vehicles. If the brightness falls below a threshold value, the low beam is activated.

With the high beam activation, the two options (HL-1/L2) differ. The simple control (HL-2) activates the high beam as soon as the user presses the corresponding button on the dashboard. For the adaptive headlight system (HL-1),

⁶ the "gateway" is the simulation connection

the simulation environment also provides an object list, allowing the high beam to be turned off when needed. Commonly, object lists are created by sensing the vehicle's surroundings using sensors such as cameras and radar, and merging this data. Detected objects are uniformly described in the object list by position, movement, and classification information (e.g., vehicle, pedestrian). In the simulation environment, the perception of the dynamic objects is an optimal ground truth, i.e., the object list includes all (simulated) road users in the vicinity of the simulated vehicle, and their position and movement are exact. As soon as objects are nearby in front of the vehicle, the headlight controller service switches off the high beam by sending the respective request via the gateway host to the headlight power actuator. A ROS 2 parameter configures the distance at which the high beam is dimmed.

Automated driving functionality

Furthermore, two options for automated driving are available. The first option (AD-1) obtains optimal waypoints from the simulation environment to reach a specified destination. AD-1 is therefore strictly linked to the simulation environment. To follow the calculated path, it uses a PID⁷ controller.

The second option (AD-2) uses the ADORe⁸ stack. ADORe is an open-source automated driving system which includes decision making, planning, and control capabilities, which was originally initiated by the German Aerospace Center (DLR) and has since been transferred to the Eclipse Foundation. The system uses map data to calculate a global route. For local control, a trajectory is calculated (depending on traffic participants) and followed by means of a PID controller. A perception solution is not part of the ADORe stack, so environmental information must be provided elsewhere. As with the adaptive headlight system (HL-2), the object list information from the simulation environment is therefore used for AD-2. In contrast to the first option (AD-1), the ADORe stack was developed for

⁷ Proportional–integral–derivative

⁸ Automated Driving Open Research (cf. glossary, page 323)

use in real vehicles. In the AutoDevSafeOps project⁹, it was used as the research object, running in a Mercedes EQV, for showcasing modular software updates and safety monitoring.

11.3 Applying the procedure to specify cybersecurity monitoring requirements

The necessary monitoring measures for the example system are identified by applying the procedure to specify cybersecurity monitoring requirements (cf. Chapter 8). The resulting monitoring concept is implemented within the miniature vehicle and the simulated fleet.

11.3.1 Threat analysis and risk assessment with threat information

Threat and damage scenario identification

The application of STRIDE to the adaptive headlight system yields several threat scenarios, of which a subset is outlined in the following. Potential threats include tampering (T) of the headlight controllers lamp request (ν_1), spoofing (S) of the lamp request (ν_2), spoofing (S) of the illumination input signal to the headlight controller (ν_3), and denial of service (D) of the headlight controller by flooding the bus with messages (ν_4) (cf. Figure 11.5). Spoofing or tampering the lamp request or spoofing the illumination signal may lead to a critical damage scenario: unintended turning-off of the headlamp system could lead to a front collision if caused during night (δ_1). In the denial of service scenario, the headlamp would remain at the latest state (i.e., low or high beam) always, which would be a less severe damage scenario (δ_2). The damage scenarios are the root of the subsequent attack trees in the attack path analysis (cf. orange-lined boxes in Figure 11.5).

⁹ cf. page 25

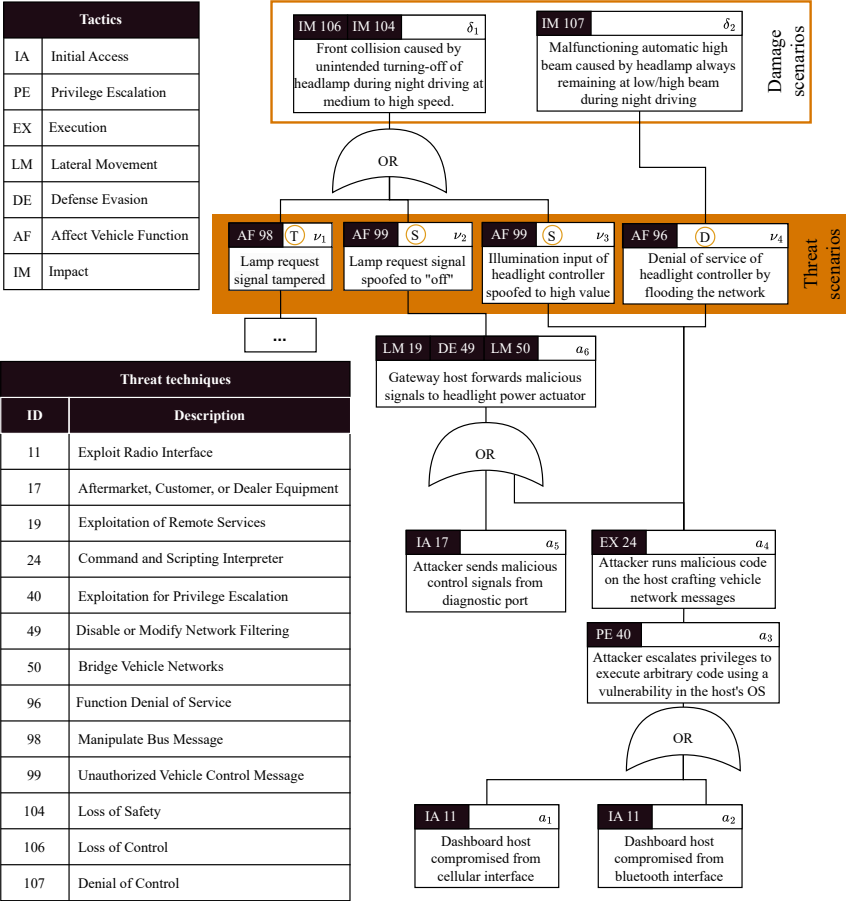


Figure 11.5: Attack tree formalization of the attack path analysis for the exemplary item.

Below in the tree (cf. orange-filled boxes in Figure 11.5) are the respective threat scenarios.

Attack path analysis

Attack path analysis aims to identify routes from entry points to triggering specific threat scenarios. The tactics "Initial Access" and "Manipulate Environment" are used to analyze possible initial attack steps. These include the steps "Exploit Radio Interface (IA 11)" and "Aftermarket, Customer, or Dealer Equipment (IA 17)", which are applicable techniques to realize initial attack steps in the example item. In this case, the external interfaces (steps a_1 and a_2 , IA 11: Bluetooth and Cellular to the dashboard host via the TCU) and the diagnostic port (step a_5 , IA 17) serve as the starting points (cf. Figure 11.1, Figure 11.5).

After accessing the dashboard host (steps a_1 and a_2), the adversary needs to ensure that manipulated signals can be transmitted. One potential path involves exploiting the dashboard host's OS to gain code execution and network capabilities, which involves privilege escalation using a further vulnerability (step a_3 , PE 40). The attacker can then execute code locally, in this example using the "Command and Scripting Interpreter" technique (step a_4 , EX 24), such as the *shell* command line under Linux. The attacker can attempt to transmit malicious control signals from the compromised host. Then, there are two subsequent attack directions: targeting the headlight power actuator directly via the gateway, or targeting the headlight controller.

Considering the first option, as the gateway host incorporates a filtering function, initially malicious messages from the dashboard are prevented from passing to the headlight power actuator. Thus, compromising the gateway's filtering functionality could be another subsequent step (a_6). The example assumes that the gateway host serves an application that can be called remotely that has a vulnerability. This could be the onboard diagnostic server, for example, which forwards diagnostic queries from connected diagnostic testers to the corresponding control units. Exploiting this vulnerability with a specifically crafted command could

allow the adversary lateral movement to the gateway host (LM 19). Modification of the gateway filtering (DE 49) permits signals to be forwarded to the following bus system. Once the gateway's filter is manipulated or bypassed, the malicious signals sent by the attacker propagate to the power switch actuator ("Bridge Vehicle Networks", LM 50), which can trigger spoofing the lamp request signal to "off" (ν_2). Alternatively to attacking from a compromised dashboard host (a_1 / a_2, a_3, a_4), an attacker gaining access via the diagnostic port might also be able to send malicious signals to the power actuator (a_5).

For the second option targeting the headlight controller, the attacker may flood the network, potentially causing a denial of service (ν_4 , AF 96). Alternatively, the attacker may send a spoofed illumination input to the headlight controller (ν_3). As this message impairs the vehicle control functionality, it is classified as "Unauthorized Vehicle Control Message" (AF 99) technique of the "Affect Vehicle Function" tactic.

Indirect manipulation (via the headlight controller) or directly spoofing the lamp request to *off* triggers the damage scenario *unintended turning-off of headlamp*. It is categorized as "Loss of Control" (IM 106), because the driver loses control of a function, and "Loss of Safety" (IM 104), because it is a safety-critical impact. The denial of service damage scenario is classified as "Denial of control" (IM 107), as the light status cannot be changed anymore by the driver.

Result of the analysis are the following seven attack paths:

- π_1 : $a_1, a_3, a_4, a_6, \nu_2$
- π_2 : a_1, a_3, a_4, ν_3
- π_3 : a_1, a_3, a_4, ν_4
- π_4 : $a_2, a_3, a_4, a_6, \nu_2$
- π_5 : a_2, a_3, a_4, ν_3
- π_6 : a_2, a_3, a_4, ν_4

- π_7 : a_5, a_6, ν_2

Risk value determination

The risk values of particular attack paths are determined by combining an attack feasibility rating for the attack paths and an impact rating for the damage scenarios (δ_1, δ_2) .

Table 11.1: Risk value calculation for identified attack paths. (-) indicates that the damage scenario is not a result of a particular attack path.

Attack path	Attack feasibility rating	Impact rating		Risk value
		δ_1	δ_2	
π_1	Medium	Severe	-	4
π_2	High	Severe	-	5
π_3	Medium	-	Moderate	2
π_4	Very Low	Severe	-	2
π_5	Low	Severe	-	3
π_6	Low	-	Moderate	2
π_7	Low	Severe	-	3

Following the example calculations given in ISO/SAE 21434, the attack paths originating from the cellular interface to threat scenarios ν_2 and ν_3 pose the highest risk (value 4 and 5, cf. Table 11.1), because the resulting damage scenario has a *Severe* impact. The attack paths toward denial of service scenario ν_4 have a risk value of two. In a complete development process, suitable measures to reduce the risks would be selected and implemented on this basis, which is not discussed further here.

Monitoring goal definition

Given the attack path π_2 has a risk value of five, the proposed monitoring goal approach (cf. Table 8.1) specifies that all attack steps should be observable. Attack path π_1 has a risk value of four, indicating that the penultimate attack step should be observed, besides threat scenario, entry point and damage scenario (cf. Table 8.1).

Given the overlap of π_2 with π_1 , monitoring goals for the penultimate attack step of p_{i1} , and for the threat scenario ν_2 need to be added. The other attack paths require monitoring of the diagnostic port as the initial attack vector, as well as the denial of service threat scenario, as their risk is lower. As a result, nine monitoring goals were defined (cf. Table 11.2).

In this example, monitoring goals address nine of the twelve elements of the attack tree (attack steps, threat- and damage scenarios). In practice, considering all threat scenarios and attack paths, the attack trees would be larger than in the example. However, since not all attack paths are high-risk, the ratio of goals to tree elements is likely to be lower for real systems, which reduces monitoring effort compared to system size. Furthermore, overlapping attack paths of several items could lead to a further reduction.

Table 11.2: Identified monitoring goals for the adaptive headlight system item.

Number	Attack step	Monitoring goal
g_1	a_1	Communication with external entities via cellular interface shall be observed.
g_2	a_3	Attempts to escalate privileges on the dashboard host shall be observed.
g_3	a_4	Attempts to execute scripts on the dashboard host should be observed.
g_4	ν_3	Abnormal transmissions of the illumination signal from the dashboard host shall be observed.
g_5	δ_1	Collisions and other critical situations when the headlamp is turned off shall be observed.
g_6	a_6	Uncommon signals passing the gateway toward the headlight power actuator shall be observed.
g_7	ν_2	Transmission of lamp request messages from the gateway shall be observed.
g_8	a_5	Uncommon commands from diagnostic port shall be observed.
g_9	ν_4	Attempts for targeted flooding of the headlight controller shall be observed.

11.3.2 Monitoring concept definition

To define a monitoring concept that translates the monitoring goals into concrete technical monitoring requirements, the mapping of techniques to observables is applied (cf. Figure 11.6). In the following, only the attack path π_2 is considered to illustrate the monitoring concept definition.

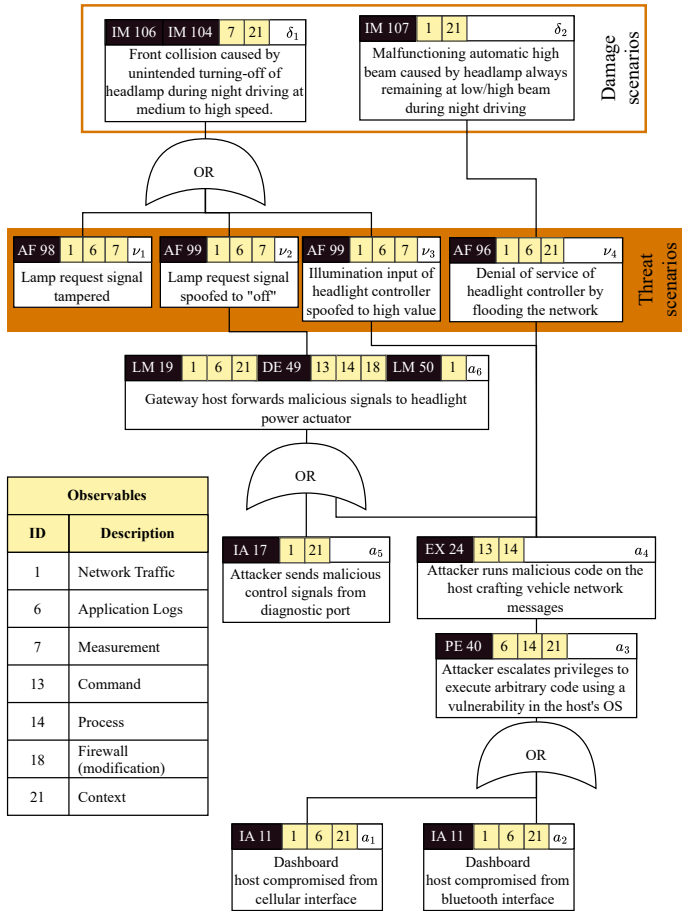


Figure 11.6: Attack tree with potential observables per attack step.

Observability determination

In π_2 , the first step to exploit the cellular interface (a_1) may manifest in the context of the vehicle (o_{21}), particularly the connection status to external networks. From a functional perspective, it is necessary to know whether there is an internet connection and, if so, whether it is with a (private) WiFi or (public) cellular network, as this may regulate the download of updates, for example. From a cybersecurity perspective, an attack step "exploit radio interface" can only occur if a connection to a wireless network has been established or a connection attempt is taking place, yielding also network traffic (o_1). Moreover, successful exploits can cause instability or unusual behavior of the abused application, which may reflect in the application's logs (o_6).

Table 11.3: Mapping of techniques to cybersecurity monitoring sensors for the example attack path.

Attack step	Observable	Sensor
a_1	o_1 Network traffic (flow and content)	Flow collector NIDS Firewall
	o_6 Application logs (content)	Application / OS logs (wireless stack)
	o_6 Application logs (content)	Application logs
a_3	o_{14} Process (creation)	HIDS
a_4	o_{13} Command (execution)	HIDS OS logs
	o_1 Network traffic (flow and content)	Flow collector NIDS
ν_3	o_6 Application logs (content)	Application (Headlight controller)
	o_7 Measurement (alert)	SIDS
δ_1	o_7 Measurement (signal)	Telemetry

Next, escalating privileges on the dashboard host's OS (a_3) may induce different observables, including process creation (o_{14}), and logs of potentially unstable OS applications (o_6). The application's or hosts status, which are other aspects of context, can be further indicators of exploitation attempts (o_{21}). Then, sending malicious control signals through execution of a command interpreter (a_4)

requires process creation (o_{14}). Then, the command is executed, which may be observable as well (o_{13}).

Finally, the last steps, spoofing the illumination signal (ν_3), can be observed in the network traffic (o_1), and particularly in signal(s) within the network traffic ("Measurement" o_7 , e.g., by SIDS). The damage scenario δ_1 essentially manifests in operational context. Particularly, inappropriate physical behavior or certain status could indicate loss of control and loss of safety. In addition, specific signals (Measurement), such as crash signals, airbag deployments, etc. could also be important observables.

After determining relevant observations to monitor the attack path, suitable sensors are identified. The technical platforms used for the hosts should be taken into account, as well as differences between the capabilities of CMS. For example, HIDS may have different technical limitations: one product may track file access, while another can track process creations. The technique-to-sensor mapping can then be summarized in a table (cf. Table 11.3, exemplary mapping of generic sensors to observables, not accounting for concrete products). Note, that collecting context is already covered by the centralized operational context monitor (cf. Section *Operational context monitoring* on page 238), not by specific CMS.

Requirements allocation

Focusing on the attack path π_2 , due to its risk value of five (cf. Section 11.3.1), it is recommended to observe each attack step with a monitoring sensor (cf. Table 8.1). As the analysis has shown, there are some steps in which several options are available (cf. Table 11.3). This is where the cost-benefit analysis and optimization comes into play (cf. Section 8.2.4), which will not be discussed further here. In sum, five requirements for dashboard host, gateway, TCU power switch actuator, and (an assumed) telemetry system were defined (cf. Table 11.4). The practical implementation in the miniature vehicle does not cover r_4 and r_5 . However, future works could include established SIDS approaches, e.g. [Web19]. In addition, several aspects of the operational context are relevant for the considered attack

path of the item, namely the external network connection status, the dashboard host and application status, and the physical behavior such as crash events.

Table 11.4: Resulting monitoring requirements and allocation to components in the exemplary architecture.

Number	Monitoring requirement	Allocated to
r_1	Deploy a flow collector on the TCU to trace external communication with the dashboard host	TCU
r_2	Log the dashboard host's operating system for errors of kernel or network application / driver	Dashboard
r_3	Deploy a HIDS to monitor for process creation and command execution	Dashboard
r_4	Deploy a SIDS for the illumination signal to observe anomalous changes	Gateway
r_5	Collect raw crash-related signals	Telemetry system

11.4 Holistic monitoring of the adaptive headlight system

11.4.1 Monitoring technology

Centralized information management

One aspect of the proposed holistic monitoring concept is the centralized intermediate storage of relevant information in the vehicle (cf. Section 9.3.1). Several tools could realize a CIM component, including syslog-ng, rsyslog, or FluentBit. However, none of the tools offers an interface to ROS 2, which is required to integrate with the developed vehicle functions. ROS 2 currently supports the languages C++ and Python. Of the tools listed, syslog-ng allows to integrate custom plugins (sources, destinations and parsers/filters) in Python, which is why syslog-ng was chosen for this prototypical implementation. As the core functionality of syslog-ng is the transmission and filtering of messages of the syslog protocol, the syslog protocol was also used as a standardized transmission format

in the vehicle and to the KG. The developed CIM service adds the VIN¹⁰ assigned to the vehicle as a key-value pair to each message before it is sent to the KG to allow to associate monitored data to particular vehicles.

To implement the extended, developer-centric logging functionality (cf. *b*) in Figure 9.6), a new ROS 2 message type was developed that mimics the syslog format. A specific syslog-ng plugin (ROS 2 source) receives the messages sent with this message type from other ROS 2 software components in the vehicle. In addition to plain text information, syslog's key-value pairs allow to send structured data, which is used, for example, by the context monitoring component.

Another plugin for syslog-ng realizes the CIM's filter buffer functionality (cf. Section *Normalization and filtering* on page 185). Featuring a ROS 2 interface, it has a configurable log level, corresponding to the severity value of a syslog message. A message whose severity is greater than the configured log level (cf. description of the syslog format, Section *Protocols and standards* on page 89) is placed in the filter buffer, i.e., it is currently not deemed relevant (cf. Figure 11.7a). The storage of the filter buffer is implemented as a ring buffer, overwriting the oldest messages as soon as the buffer fills the reserved memory. All others are given to the output buffer¹¹ to be forwarded to the VSOC as soon as connectivity is established, i.e., these messages are currently considered relevant.

When a ROS 2 request to change the log level comes in, the severity of all messages in the filter buffer is compared against the new log level (cf. Figure 11.7b). If it is below the changed log level (more severe), it is placed in the output buffer to be sent to the VSOC. Otherwise, the message remains in the filter buffer.

All other information from CMSs, in particular ordinary textual logs from ROS 2 containers, network flows (cf. Section *Firewall and network management* on page 240) and alerts from the HIDS system (cf. Section *Host-based intrusion*

¹⁰ Vehicle Identification Number

¹¹ syslog-ng natively ships with a reliability feature, including buffering of outputs in case of missing connectivity.

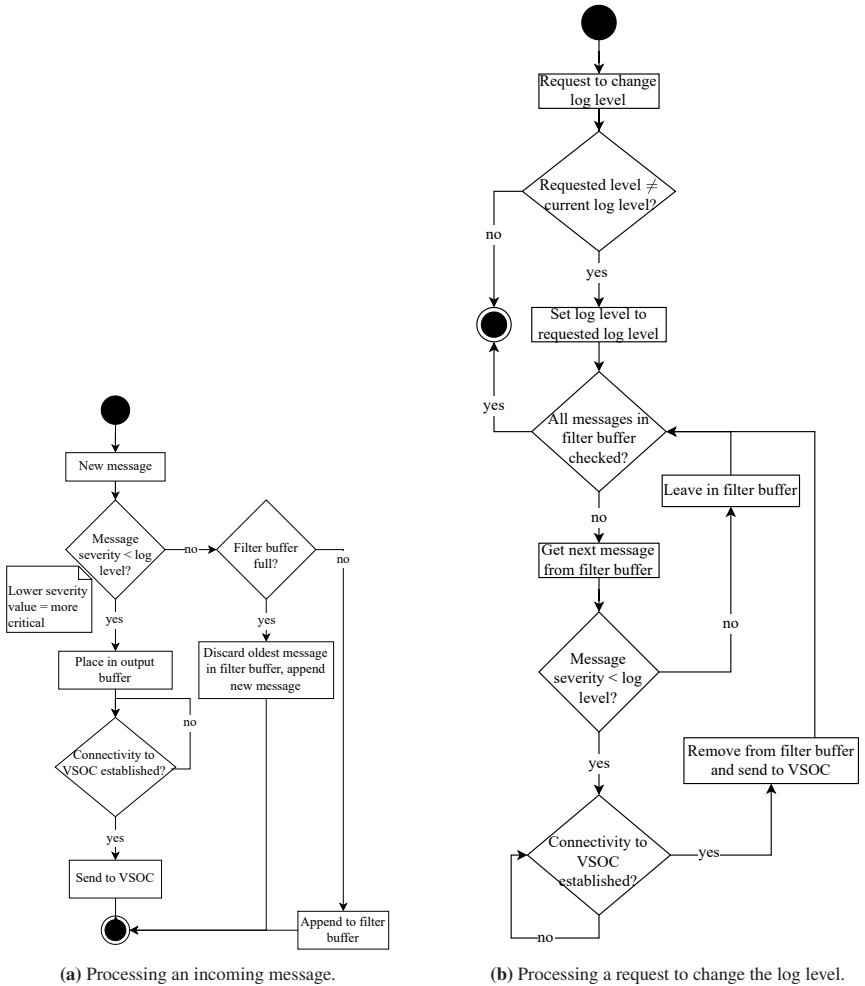


Figure 11.7: Activity diagram of the filter buffer in the CIM implementation.

detection on page 241) are sent to the CIM via TCP using the syslog protocol and processed according to their severity value.

Operational context monitoring

A ROS 2 service was developed to monitor the operational context which is sent in regular intervals to the CIM as a ROS 2-syslog message. The service is structured according to the context aspects defined in ISO 34503 and ISO 34504, i.e., environment, scenery, behavior and dynamic elements, plus the temporal aspect (cf. Section 9.2.2). Each aspect is implemented as a separate module¹² that calculates the respective information using different data sources. The calculated aspects are then combined¹³ into a single context message and sent to the CIM.

a) Environment The environmental conditions include aspects of lighting, weather and connectivity. For cybersecurity, the connectivity aspects are important particularly to analyze initial access vectors of attack paths. In addition, for the adaptive headlight system, the illumination conditions are particularly relevant. As the wireless connectivity of the prototypical implementation is limited to WiFi, only the connection to certain servers and the public internet is considered. For the weather and lighting, the implementation conforms to the ISO 34503 standard's recommendations for classifying measured values into categories.

b) Scenery The scenery context module focuses on the identification of geographical zones, road types and administrative environments based on the current position (latitude, longitude). Since the legislative requirements depend on national borders, these are of particular importance. In addition, the first highly

¹² The term *module* refers here to different execution contexts, so-called "ROS 2 nodes", cf. <https://docs.ros.org/en/iron/Concepts/Basic/About-Nodes.html>.

¹³ In the future, consistency checks of the different aspects (environment, scenery, dynamic elements, time) of the operational context or of the vehicle in relation to its context could be integrated to improve context handling, particularly for automated driving functions.

automated driving systems were released on highways, which is why the identification of such context is relevant. Activation of an adaptive headlight system, for example, may be restricted to rural areas.

For the administrative boundaries, a Python package based on the database *GeoNames* determines the nearest city and associated region in a tree structure optimized for fast queries. Besides, public map data from *OpenStreetMap*¹⁴ which is downloaded for the current region, serves to identify road types (e.g. freeway, country road, etc.), speed limits, as well as relevant areas and zones (residential area, industrial area, etc.). Offline storage of small map regions means that there is no need for a continuous internet connection and faster query times.

c) Behavior and dynamic elements Observing the behavior of surrounding road users is a central capability of modern vehicles. For example, object lists are required also for the adaptive headlight system. Particularly, automated driving functions require comprehensive understanding of the environment to plan safe trajectories.

Object lists are a common granular output of perception software stacks (obtained from simulation in the present implementation) that represent concrete observations of the environment. Beyond granular information, the implemented dynamic elements module extends to an abstraction of the dynamic elements in relation to the vehicle. Using ADORé's map library, matching the positions of all other traffic participants to lanes and roads, and comparing them with the vehicle's position (in front, behind, etc.) yields the dynamic elements' operational context from the object list.

d) Time The time context module yields a set of temporal information, using the previously obtained information about the exact location (state and country). Particularly, the local time is calculated from the global world time by lookup of

¹⁴ Collaborative, crowd-sourced map service

the timezone¹⁵. A public database is queried¹⁶ to retrieve if the current day is a public holiday, or workday.

11.4.2 Further in-vehicle software

Firewall and network management

The firewall software *pfSense* [BUR06] is used to control external access to the vehicle. This software (which is free as an open-source version) was selected due to its widespread industrial use, ease of configuration, and wide range of functions. Controlled external access to applications in the vehicle can be enabled using port forwarding¹⁷. Requests for a specific public UDP or TCP port are forwarded to a host in the vehicle network. The necessary firewall rules were configured via a graphical user interface to allow, for example, the VSOC accessing the incident response controller, but to block all other external connection attempts. In addition to classic firewall functions, *pfSense* is used for network management, as DHCP server, DNS resolution and forwarding, and the WireGuard VPN solution are integrated into it. The plugin *softflowd*¹⁸ is used to monitor the vehicle's network connections to the Internet or to private servers via the VPN (cf. requirement r_1 in Table 11.4). *Softflowd* tracks packets and bytes for all network flows and automatically informs the CIM.

Using an additionally developed interface, status information can be retrieved via ROS 2 (e.g., connectivity to the update server, public internet connection). Besides, the incident response controller uses this interface to activate or deactivate certain firewall rules. This allows various response actions to be implemented, in particular for containment (cf. Table A.8).

¹⁵ <https://github.com/jannikmi/timezonefinder>

¹⁶ <https://github.com/workalendar/workalendar>

¹⁷ cf. glossary, page 327

¹⁸ <https://docs.netgate.com/pfsense/en/latest/recipes/netflow-with-softflowd.html>

Host-based intrusion detection

In addition to network flow information and application logs, a host-based intrusion detection system was selected as part of the monitoring strategy (cf. requirement r_3 in Table 11.4). The *falco* tool [The24] is widely used for containerized software deployments and is anchored in the CNCF. As such, it is deployed on the host *gui* and configured so that all alerts are sent to the CIM as syslog messages¹⁹. With a focus on the implemented attack path, a rule was also written that is capable of detecting the initiation of the implemented attack path (cf. reverse shell, Section 11.5.1).

Software allocation and status

Docker is used as the container runtime for deploying any developed, containerized software on the hosts (*gui*, *center*, and *wing*). The specific set of containerized services running on each host was defined using Docker's Compose files. All containers to be executed can thus be automatically downloaded from GitLab's CI/CD system and started.

To monitor the status and provide the deployment information required for the KG, a service was installed on each host that regularly queries the container runtime. Syslog messages are then sent to the CIM via the extended logging interface (ROS 2), informing about the currently running containers. In addition, the service provides status information about the host, such as memory consumption and CPU usage.

11.4.3 Fleet data collection in the Knowledge Graph

The prototypical realization of the KG builds on the graph database *Neo4j*²⁰ (cf. Section *Database technologies* on page 108), which is written in Java and widely

¹⁹ Using the *falco-sidekick* tool was required for this purpose

²⁰ <https://Neo4j.com/>, used version: Neo4j enterprise 2025.10

used across different industries. The Neo4j graph database can be accessed by writing queries in Neo4j's *Cypher*® language. Queries are executed via a browser-based command line interface (mainly for development and debugging purposes), or using libraries for various programming languages (including Java, Python, Go, and .NET). The latter is typically used to develop applications that intend to integrate a Neo4j database. Another option to work with a Neo4j database is Neo4j Desktop²¹, which is a tool providing graph visualization and exploration features beyond the browser-based command line. These features allow to interactively analyze the graph's data with elaborate layout and styling functions without having to build a custom application. While Neo4j Desktop served to analyze the implemented attack path (cf. Section 11.5.2), the Python library was used to prepare the graph from documents, and ingest data from the fleet.

Preprocessing and integration of development process documents

The KG implementation involves of a set of scripts to parse and load prepared documents regarding the development process (cf. Section 9.1.2 on page 170). The documents include information that is associated with particular configuration(s), such as variant management, logical architecture and risk analysis. It also includes information independent of particular vehicles such as threat information and analysis and handling views. Further scripts were developed to initialize an E/E architecture for the three realized vehicles in the KG. All initialization scripts are written as Python code that runs several separate Cypher queries. As a result, the KG after initialization with the development documents (cf. Figure 11.8) contains a vehicle shadow view per vehicle, including links of the E/E architecture of each vehicle to the vehicle-independent development view, and the development and threat information views among each other.

²¹ <https://neo4j.com/docs/desktop/>, used version: 2.0.5

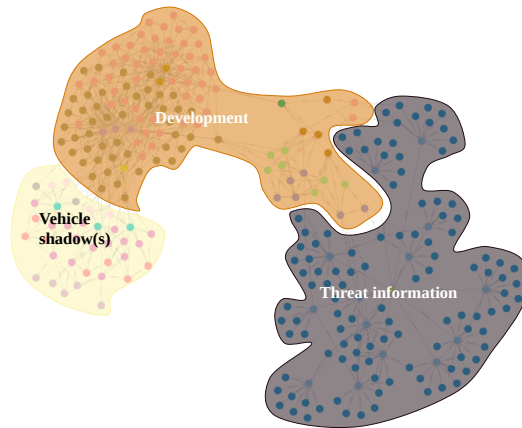


Figure 11.8: Knowledge Graph after initialization with development documents. Note: color scheme of nodes automatically determined by Neo4j, overlay for views matching the thesis' scheme.

For example, the following Cypher statement is used in the script to instantiate a configuration in the graph, linking the variant management to the logical architecture within the development view:

```
1  MATCH (c:Configuration {name: $variant_name}) -[:HAS_FEATURE]->
    (f:Feature {name: $feat_name, revision: $feat_revision})
2  MATCH (comp:Component) -[:REALIZES]-> (f)
3  MERGE (c) -[:HAS_COMPONENT]-> (comp)
```

First, the query finds the previously created configuration (MATCH) and the features that are part of it (1), then finds all components that realize the found features (2), and finally assigns a relationship HAS_COMPONENT, stating that the configuration involves the found components (3). The MERGE statement only creates a new relationship, if it is not yet present in the graph, which ensures that the initialization scripts can be re-run without creating a new relationship every time.

The variant, logical architecture, and risk documents were defined as YAML²² files. While YAML has a limited cross-referencing functionality compared to industry-standard XML files, it has a benefit of both human and machine readability, which enabled to specify the architecture without specialized tooling. In the future, document exports of industrial-proven tools should be used in place of these files. For the threat information view, the STIX-compliant JSON file defining the ASCOT matrix (cf. tool implementation, Section A.2 on page 281) was parsed using a specialized Python library²³ (cf. Figure 11.9b). As the risk analysis was carried out involving the proposed approach for monitoring concept definition, all attack steps were linked to the respective techniques of the ASCOT matrix in the KG (cf. data model in Section *Development* on page 194).

As an example, the attack tree for a particular item can be shown using the KG (cf. Figure 11.9a, relationships to techniques not shown) with the following query:

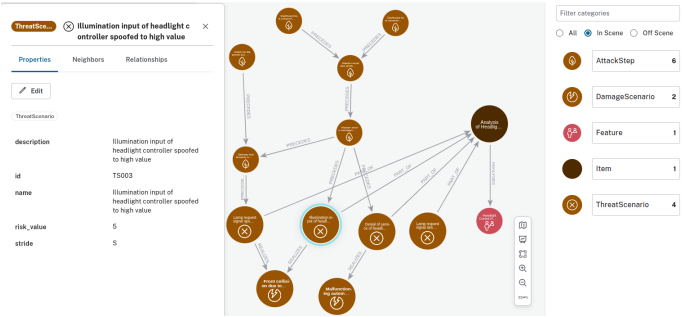
```
1 MATCH (f) <-[r:ANALYSES]- (i:Item) where elementId(f) in $nodes
2 MATCH (i) -- (t:ThreatScenario)
3 MATCH (t) -[r2]- (k:DamageScenario)
4 MATCH (t) -[r3]- (k2:AttackPath)
5 MATCH (k2) -[:CONSISTS_OF]- (a:AttackStep)
6 MATCH (a) -[p:PRECEDES]- (a2)
7 RETURN a, p ,a2, r2, k, i, r, f
```

Here, `$nodes` is a set of nodes, that the user clicked in the interactive graph view. For the query to return an attack tree, the clicked node(s) must be a component or feature, that is analyzed in a TARA item. If the relationship `[ r : ANALYSES ]` does not exist (first line of the query), there is no item related to the selected component or feature.

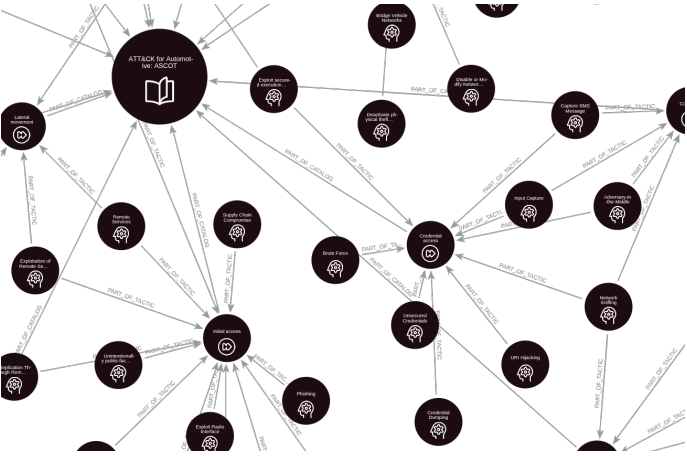
Beyond TARA documentation, for the development view (cf. Figure 11.10), feature models were defined for variant management and realized as two documents for the three considered vehicles (cf. Figure 11.10a). Particularly, the two simulated vehicles (B/C) stem from the same system revision, but select a different set

²² Yet Another Markup Language (cf. glossary, page 330)

²³ <https://mitreattack-python.readthedocs.io/>



(a) Attack tree in the KG.



(b) ASCOT matrix in the KG.

Figure 11.9: Implementation of TARA and relationship to the threat information view in the KG.

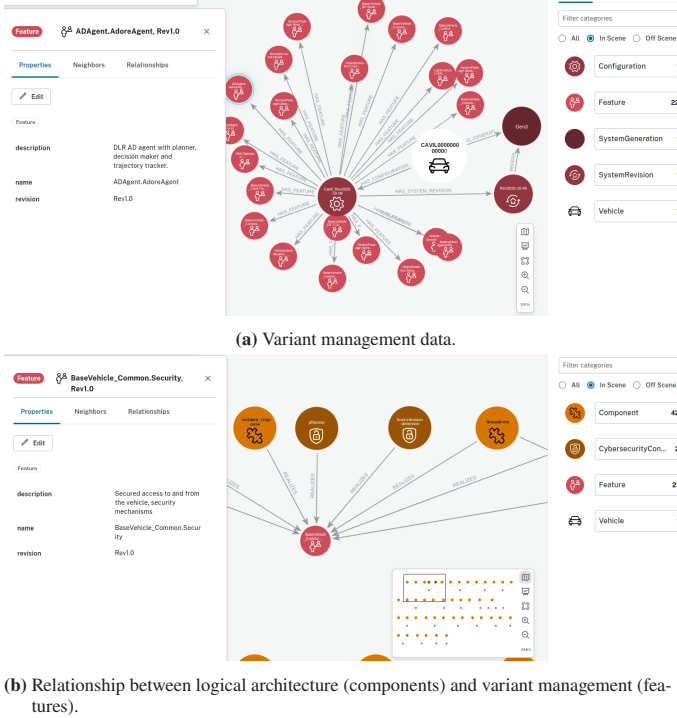


Figure 11.10: Implementation of the development view in the KG.

of the 33 specified features. As such, a common document defines their features and dependencies. The miniature truck's configuration (vehicle A) corresponds to the subsequent system revision, as it involves an update of the infotainment system. Note, that a feature model as a means of variant management usually captures variable product characteristics in general. As such, variability includes both customer-visible characteristics (in the original sense of *Feature*, cf. Definition 2.4) as well as other production-relevant variability not directly selectable by a customer.

The logical architecture is defined with a file per system revision, i.e., two documents for the considered example. The logical architecture documents define the version of each logical component and which feature (and feature revision)

it implements (cf. Section *Development* on page 194). As a result, the features contained in a system revision are realized by one or more of the 48 defined components (cf. Figure 11.10b). Software components (35 of the 48 components) are annotated in particular with the respective repository where the code resides, enabling to directly navigate to the source code from the graph later on. Data and capabilities were not realized in the prototypical implementation.

Moreover, using GitLab’s CI/CD system, vulnerability scanning of the developed containerized software components was performed with the tool *Trivy*²⁴. This step generated an SBOM in the industry-standardized CycloneDX²⁵ format (version 1.6, as JSON file) for each component, including identified vulnerabilities of the software installed in the respective container. The preprocessing scripts parsed the SBOM files, building up the relationships of the (containerized) software components specified in logical architecture documents with their sub-components (development view), and assigning a *WEAKNESS* for each found vulnerability (cf. Section *Analysis and handling* on page 201). The weaknesses relate to nodes as references for the respective CVE (or other) database entries, as part of the threat information view. In the scanned 35 (containerized) software components, 3105 sub-components (i.e., installed software in the containers) were found, that have in sum 2938 weaknesses (CVE-referenced vulnerabilities), of which 140 have a high severity²⁶. As such, depicting the components, sub-components and weaknesses in a single figure is not possible.

In contrast to the development view, the E/E architecture is implemented in the KG in a vehicle-specific manner (as part of the vehicle shadow view) to enable unambiguous assignment of the information collected to the respective vehicle and to related software or hardware (e.g. application log of specific software in a specific vehicle). Accordingly, the vehicle shadow is initialized for each produced vehicle with the information that is not (or cannot be) sent from the vehicle. The prototypical implementation includes YAML documents for the implemented

²⁴ <https://trivy.dev/v0.67/>

²⁵ cf. glossary, page 325

²⁶ based on the CVSS score assigned to the CVE entry.

configurations, which define the installed hardware (hosts, networks, sensors) and their physical connections. For each vehicle, a script initializes the corresponding E/E elements and their relationships in the KG. The E/E architecture elements are linked to the logical components that they implement and to a `VEHICLE` node, which contains the VIN and production information. In total, the KG (including the three generated vehicle shadow views, excluding weaknesses and subcomponents) comprises 287 nodes with 25 different labels (corresponding to the classes from the data model) and 695 relationships before information from the vehicles is integrated.

Preprocessing and integration of in-vehicle information

While integrating information originating in the development process required to parse various documents, the vehicle fleet provides the major share of cybersecurity information and the operational context (cf. Section *Vehicle fleet* on page 168). A WireGuard VPN server was set up to provide secure networking of the vehicles (miniature and virtualized) with the KG. The server routes communication from the vehicle to the backend (VSOC) and vice versa. The firewall of each vehicle establishes the connection to the WireGuard server and enables access by the incident response application to the response controller in the vehicles via port forwarding.

The CIM transmits the information from the vehicles to the backend via the VPN connection. There, another syslog-ng instance receives and decodes the data sent by the vehicles. A Python plugin uses the *application name* field of the messages to generate a suitable KG insert or update statements. For example, information stemming from the operational context monitoring will create `CONTEXT` nodes, while information from softflowd results in `NETWORK TRAFFIC` nodes. The information is assigned to the vehicles using the VIN sent along with it. As such, running containers can be traced back to the host and vehicle where they are deployed (cf. Figure 11.11).

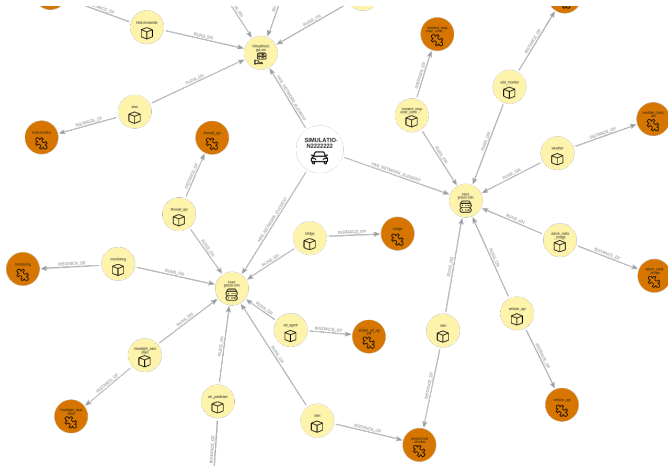


Figure 11.11: Information collected from vehicle (C): Containerized software instances linked to the respective hosts and components.

11.5 Demonstration of the incident handling workflow

The workflow with analysis in KG and incident response is illustrated using an example attack path. Similar to the previously modeled TARA, initial access, the command and control channel, and the final impact were implemented. Overall, the attack path involves a supply chain attack that allows the attack to remotely execute malicious commands in the miniature vehicle and a real vulnerability of the chosen ROS 2 middleware implementation, that is exploited remotely (cf. overview in Figure 11.12 and steps 1-3 in Section 11.5.1).

11.5.1 Implemented attack path

1. Initial access and persistence Since the used TCU is not known to be vulnerable for wireless exploitation, a supply chain attack was chosen as the initial

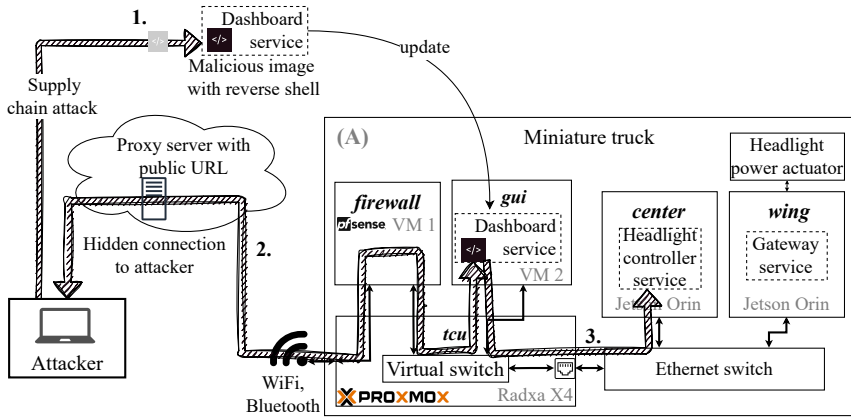


Figure 11.12: Technical view of the implemented attack path (1-3) in the miniature vehicle's E/E architecture (cf. Figure 11.2b on page 220).

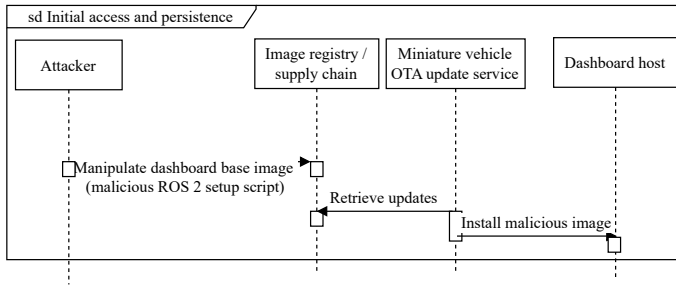


Figure 11.13: Sequence diagram for attack step 1: initial access and persistence

point of entry (in contrast to the TARA, cf. Section 11.3.1). Supply chain attacks are highly scalable, since in the worst case, all products that integrate the vulnerable software (or hardware) are vulnerable. A manipulation of the dashboard's container image realizes the supply chain attack in the given prototype (cf. Figure 11.13). With the next OTA update, the malicious image is installed.

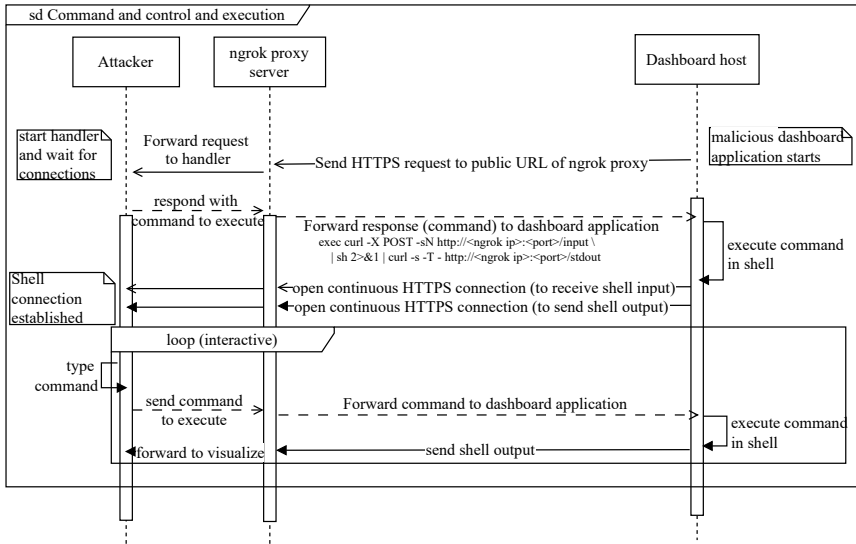


Figure 11.14: Sequence diagram for attack step 2: command and control and execution

The malicious image changes the ROS 2 setup script, which leads to execution of injected malicious code besides the original functionality²⁷ of the script. Integrating malicious code in the application's image also ensures the persistence of the attack, i.e., it is executed every time when the vehicle restarts.

2. Command and control and execution An interactive reverse shell via HTTPS²⁸ was implemented for command and control (cf. Figure 11.14), which is executed in the containerized dashboard software. Using HTTPS as command and control channel has two reasons. First, HTTPS is usually a permitted communication protocol for an infotainment system, for example, to access websites. Thus, HTTPS is actually a viable command and control channel in a vehicle. Second,

²⁷ Majorly, the setup script of ROS 2 appends all required file system paths to the system's path environment variable.

²⁸ Hypertext Transfer Protocol Secure

it is advantageous for the attacker, as it is an encrypted channel that hides the attacker's actions.

In addition to encryption, the public proxy server *ngrok*²⁹ was used to further conceal the attack. In detail, communication between the affected vehicle and the attacker proceeds as follows: first, the malicious base image's code introduced via the supply chain contacts a hard-coded, public web server address provided by ngrok. Ngrok forwards the request to the attacker's computer. There, a counterpart software (the *handler*) receives the request from the vehicle³⁰ and sends a command in response, which is executed by the vehicle. This gives the attacker interactive remote access to the vehicle, allowing to execute arbitrary commands within the containerized dashboard application.

3. Impact The final step is to exploit an existing, real vulnerability in the DDS middleware *Fast-RTPS* from *eProsima*. Version 2.13.3 of Fast-RTPS has a heap buffer overflow vulnerability that can cause DDS to terminate when a specially crafted message is sent. The official CVE entry "CVE-2024-28231" was assigned to the vulnerability after it was published in March 2024³¹. A patch has already been released, so systems currently in the field no longer have this vulnerability, provided their DDS middleware has been updated.

In the used version ROS 2 "iron", Fast-RTPS is the default DDS implementation. To exploit this vulnerability in the miniature truck (vehicle A), Fast-RTPS version 2.13.3 was recompiled, creating a vulnerable ROS 2 container image. Any ROS 2-based software that uses this image is potentially vulnerable to the attack described above. Specifically, it has been tested that any ROS 2 subscriber that subscribes to a topic with a dynamic length data type can be remotely shut down by exploiting this vulnerability. An object list, for example, has a dynamic length because the number of objects in the environment varies.

²⁹ <https://ngrok.com/>

³⁰ based on <https://github.com/scriptzteam/CurlShell>

³¹ <https://github.com/eProsima/Fast-DDS/security/advisories/GHSA-9m2j-qw67-ph4w>

Using the container image featuring the vulnerable DDS implementation, a vulnerable adaptive headlight system component was created. In addition, a script was developed that exploits the vulnerability by defining a targeted topic automatically. The vulnerable image and the script for the automated attack have been made publicly available³². The only requirement for the attack to succeed is that the publisher and subscriber must not run on the same host. If they do, the attacker must also execute the attack on this host.

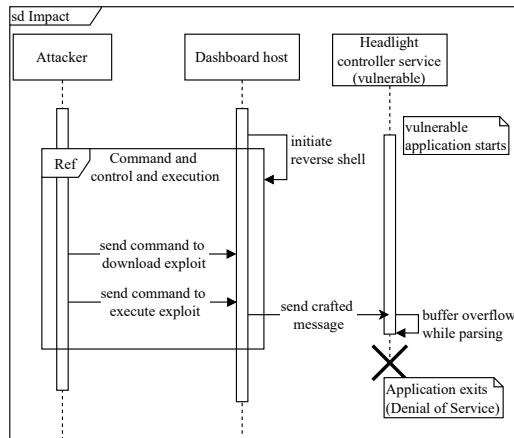


Figure 11.15: Sequence diagram for attack step 3: impact

To impact the vehicle's functionality, the attacker retrieves the script in the reverse shell via the command and control channel and executes it interactively (cf. Figure 11.15). Targeting the object list used by the adaptive headlight system to turn down the high beam, the publisher and the attacker are on the same host, while the target runs on a different host. The result is a remote denial of service, i.e., the adaptive headlight system crashes with an error message. The vehicle maintains the last light status. If the high beam was activated, the surrounding traffic is blinded.

³² <https://github.com/grimmbo/ros2.CVE-2024-28231>

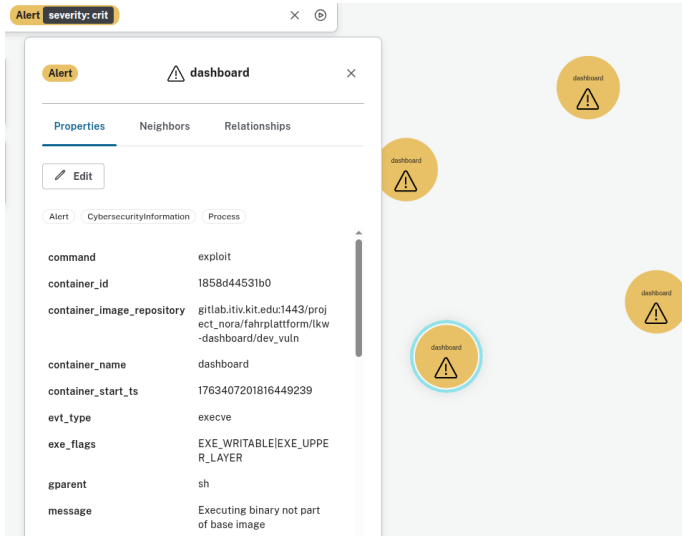


Figure 11.16: Critical alerts produced by the HIDS on the implemented attack path. Note the query shown in the upper left: searching for ALERT nodes with severity "crit" (critical).

11.5.2 Using the Knowledge Graph to analyze the attack

The initial phase of analyzing a potential attack often begins with the occurrence of an alert. This refers to information from the vehicle that has been classified as critical or otherwise particularly relevant. In the present case, the deployed HIDS on the *gui* VM (cf. VM 2 in Figure 11.12) detected parts of the attack, specifically the execution of a downloaded exploit within the container. Given this background, multiple nodes are found when querying the graph for critical alerts (cf. Figure 11.16). If the HIDS is deemed highly reliable, these alerts may already justify the creation of a security event by the analysts. In the KG, the progression of the analysis is captured with corresponding nodes (EVENT, INCIDENT, etc., cf. Section *Operations process* on page 174).

To support different analysis approaches, several Cypher queries were implemented as so-called *Scene actions* in Neo4j Desktop, that can be used with a right click on a node in the graph (cf. Figure 11.17). First, the alert's context and the

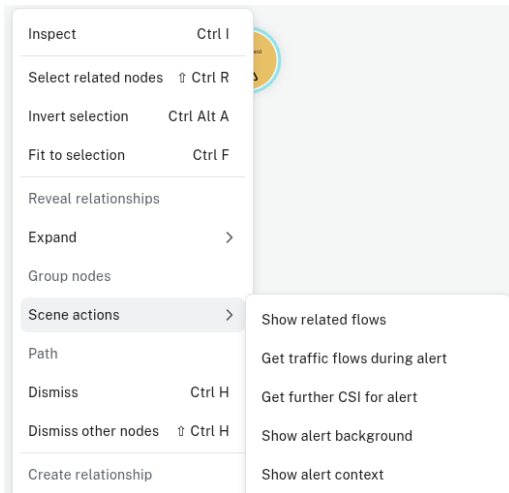
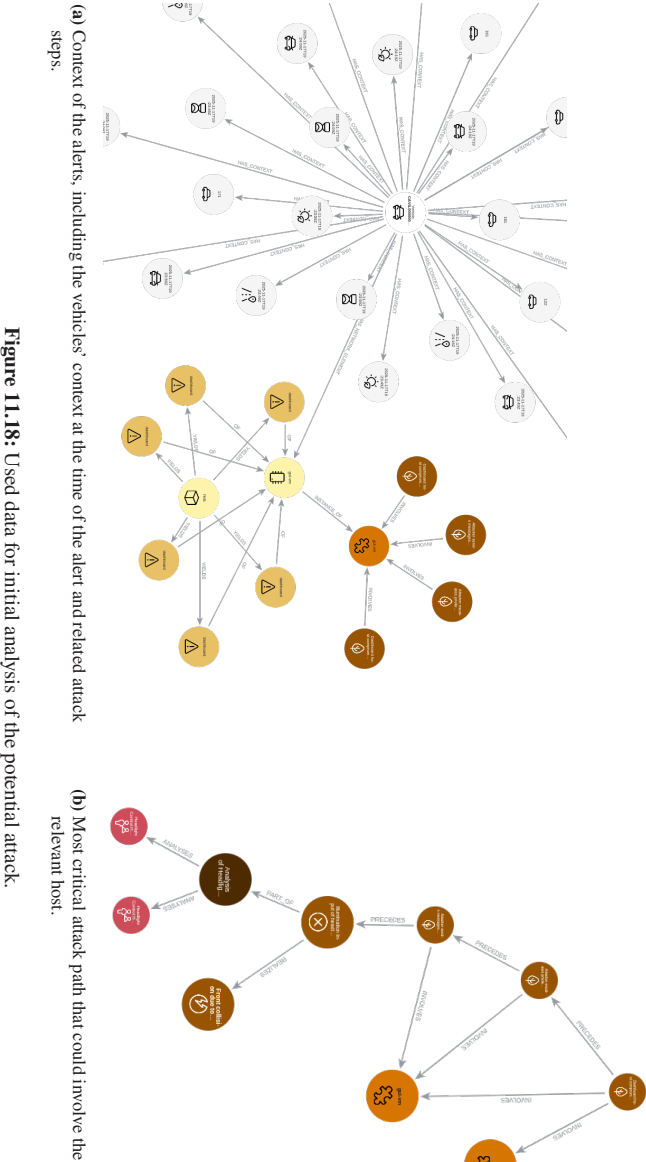


Figure 11.17: Scene actions in Neo4j Desktop to analyze alerts.

affected vehicle are analyzed. This includes assessing the potential consequences if the alert indicates a confirmed attack. The analysis focuses on two key aspects (cf. Figure 11.18a): The vehicle's current state at the time of the alert (e.g., environment, movement, and active features like light control), and the known attack steps associated with the affected component—in this case, the *gui* VM. The investigation reveals that all active alerts originate from the same host, for which multiple attack steps have been modeled in TARA. Given the vehicle is operating in a dark environment and is in motion (cf. Figure 11.19), features such as light control become particularly relevant. To evaluate the worst-case impact, the analyst examines all attack paths involving the relevant steps. The path representing the highest-risk threat scenario is identified (cf. Figure 11.18b). This confirms the initial hypothesis: the headlight assistant is linked to the alerts (via the TARA item, that analyses the respective feature), and the scenario carries the maximum risk rating of five (cf. Section *Risk value determination* on page 230). A detailed incident analysis is therefore warranted.



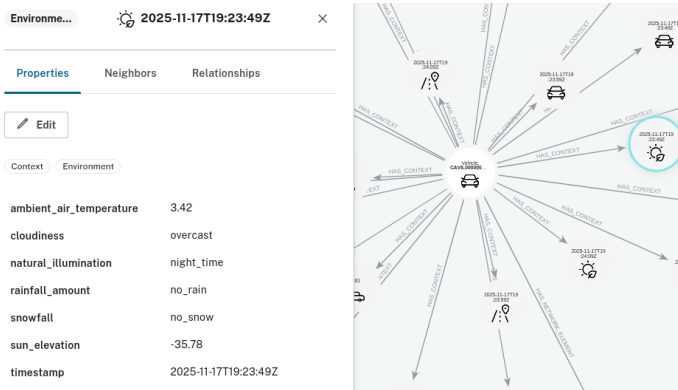


Figure 11.19: Environmental conditions of the vehicle during the alerts.

Given the critical alerts and the need for deeper investigation, a targeted incident response action is initiated. Using the incident response management application (cf. Figure 11.20), the decision to change the log level from *warning* to *informational* is prepared to gather more detailed forensic data. After connecting and sending the command to the vehicle's incident response controller, the CIM processes the filter buffer (cf. Section *Centralized information management* on page 235 and Figure 11.7b) to forward the previously kept messages that are now deemed relevant³³ due to the changed log level. This ensures that more relevant information is collected, providing a comprehensive dataset for analysis.

Upon reviewing the expanded logs, additional HIDS alerts are identified, revealing external network communication to a specific IP address (cf. Figure 11.21a). The alerts explicitly state that this communication is initiated from within the dashboard container, which raises immediate concerns about potential compromise or unauthorized activity. To validate these findings, an examination of the collected network flows confirms that the vehicle is indeed communicating with two external address over port 443 on a frequent basis (cf. Figure 11.21b). Resolving the IP addresses to a domain name uncovers something unusual: the

³³ According to the syslog standard, the level is changed from 4 to 6, i.e., messages that have a severity of 5 or 6 are now forwarded to the VSOC instead of being kept in the filter buffer

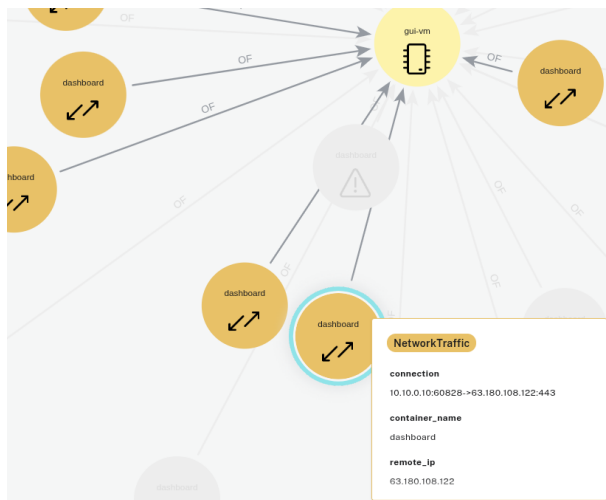
The screenshot shows a web-based interface for incident response management. At the top, it displays 'Vehicle: 10.0.0.102' with a dropdown menu showing '10.0.0.102'. Below this is a 'Stop connection' button. The 'Connection status' is 'connected'. The interface is divided into two main sections: 'action' and 'Prepared actions'. In the 'action' section, there is a dropdown menu showing 'CHANGE_FILTER_SEVERITY', followed by fields for 'valid from (UTC)', 'valid until (UTC)' (with the date '2025-11-17'), and 'additional action data (str) [optional]'. Below these are fields for 'additional action data (int) [optional]' with the value '7', and two buttons: 'add action' and 'save changed action'. In the 'Prepared actions' section, there is a dropdown menu showing 'CHANGE_FILTER_S'. At the bottom right, there are three buttons: 'delete selected action', 'load action from file', and 'Provide prepared actions'.

Figure 11.20: Incident response management application.

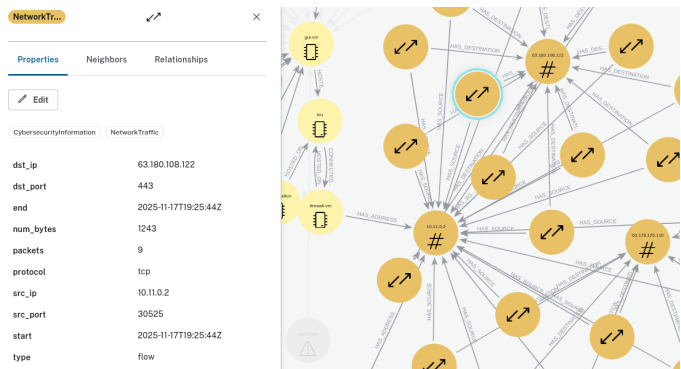
address belongs to a cloud service provider that is not used by the vehicle manufacturer³⁴. This discrepancy strongly indicates that the communication is not part of standard operations, indicating a potential security incident that requires further investigation and response.

After identifying the suspicious external communication, the analyst evaluates the status of the running component instances in the vehicle. The investigation reveals that the headlight assistant process has exited unexpectedly (cf. Figure 11.22), which is an unusual occurrence under normal conditions. Given this finding, combined with the earlier detection of unauthorized network activity and the high-risk threat scenario, the presence of an active incident is assumed. The unexpected termination of the headlight assistant aligns with the evidence of a compromise, likely tied to executing the exploit and external communication.

³⁴ The IP addresses actually belongs to Amazon's cloud (AWS), where the ngrok proxy is hosted. The fact that the manufacturer uses a different cloud provider is merely an assumption for the purposes of this demonstration.



(a) Alerts indicating external communication from within the container.



(b) Network flows showing the vehicle's external communication attempts.

Figure 11.21: Additional information collected after increasing the log level.

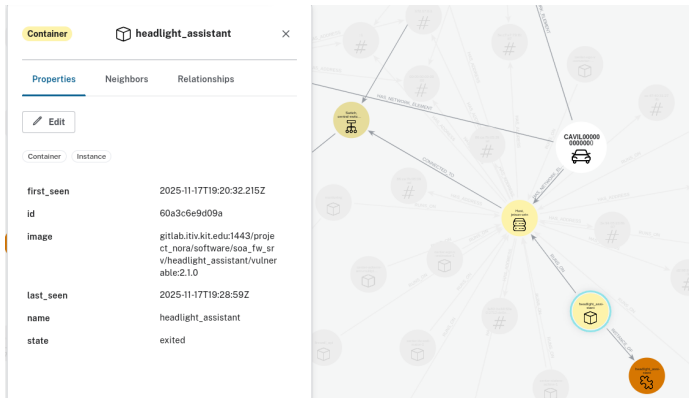


Figure 11.22: Confirmed problem with the headlight assistant using its status information.

Immediate response measures, such as containment, are now demanded to address the incident effectively. In the given example, using the incident management application, a containment action is sent to the incident response controller, which shuts down the vehicle’s external communication via the ROS 2 firewall API. The containment action hinders the attacker from transmitting its command and control messages via the public server. Afterward, the graph records no new flows to the external destination, indicating a successful interruption of the attack at the cost of limiting the customer’s experience with internet-connected applications.

11.6 Discussion of the outcomes and limitations

Operations process and improvements to ISO/SAE 21434

Due to the prototypical research context of this work, the operations process could not be implemented in practice, limiting the discussion to qualitative aspects. In general, the proposed operational process explicitly defines when and where risk analyses should be updated, when and how to carry out incident and vulnerability handling, and when and why there is a need to improve defensive measures or

monitoring mechanisms (cf. post-analysis activities). The process thus definitely goes beyond what ISO/SAE 21434 contributes to the fulfillment of the tasks from UN R155. However, vehicle manufacturers must still demonstrate the actual fulfillment of the duties in an individual certification process. Cooperation with suppliers is not a key element of the proposed process, but brokering access to TARA artifacts is a relevant contribution proposed in the process. In principle, the cybersecurity interface agreements in ISO/SAE 21434 already offer a practical approach to cooperation, although defining interface agreements to cooperate on vulnerability and incident management could be further refined in the future.

The improvements to the operations process aspects in ISO/SAE 21434 were submitted to the standardization committee of ISO/SAE, and discussions on future revisions of the standard will likely debate them. Feedback from an industry partner (IT service provider for OTA updates, vehicle software, and VSOC solutions) on the proposed process has already confirmed some of the factors. To give an example, "the terminology definitely needs to be adapted and closure & post-analysis activities are actually completely missing" (translated from German). Other aspects are not necessarily considered a vital extension, but as a valuable baseline that could complement the standard: "The incident handling process is actually already supplemented by the [UN] R155. Your suggestion could perhaps be included as a reference" (translated from German). Discussions with another industrial partner (OEM) furthermore confirmed the relevance of connecting development and operations perspectives for cybersecurity, particularly relating TARA outcomes to the vulnerability and incident handling. Besides, related research projects involving DevOps approaches, e.g., AutoDevSafeOps addressing the safety of vehicles, highlight the relevance and importance of continuous operations processes for the automotive industry in general. Overall, the ISO/SAE standardization committee's interest and the industrial confirmation show the relevance of the contributions regarding the operations process. However, proving the suitability of the proposed processes requires a long-term industrial implementation and evaluation in the future.

Procedure to specify monitoring requirements

The example from ISO/SAE 21434 (and the second example in the appendix, cf. Appendix B) showed that the method for defining monitoring requirements can be successfully applied. Implementing the selected monitoring requirements subsequently in the practical example of the miniature truck also implies that the monitoring solutions were suitable for detecting attacks in the vehicle. Although the attack implemented was not precisely the same as the one modeled, the incident could be identified. Of course, this example does not prove the universal validity of the proposed method. However, it does show that integration of the approach into the existing TARA methodology is feasible. Compared to development without consideration of monitoring requirements, additional effort is required to identify structured requirements.

At the same time, mapping attack steps to techniques provides methodological support for TARA that goes beyond its mere application for identifying monitoring requirements. For the continuity of the cybersecurity operations process, it is essential to define monitoring requirements and review them regularly by evaluating the monitoring results, regardless of how the development process establishes these requirements. However, the ease of integrating the proposed procedure into a vehicle manufacturer's standard development process requires thorough evaluation. Particularly, the evaluation of supplier products as CMS designed to detect observables, could bear initial difficulties. Furthermore, an industrial assessment should evaluate the additional effort, especially in comparison to statistical methods (cf. Section 4.1.4).

Holistic monitoring

The proposed holistic monitoring approach, as demonstrated in the above example, integrates development and operational data within a KG-based data model, aligning with SPLE-based MBSE artifacts (cf. Section *Variant and release management* on page 30) to ensure consistency across vehicle variants and close the

lifecycle loop between development and operations. Combining development-time artifacts such as architectural models and threat analyses with real-time operational data, the approach enables enhanced, context-aware, and cybersecurity situation-aware analysis of cybersecurity information and events. This integration reduces manual effort to correlate disparate data sources, accelerating incident analysis and response. In the presented example, the environmental conditions and status of the components were highly relevant for assessing the situation. Also, the graph facilitated straightforward correlation of the network flows with the vehicle's addresses. Without the central feature, a KG to represent relationships among alerts, system components, and contextual information, alerts would remain isolated, complicating the assessment of their broader implications and the prioritization of responses. In the second example, discussed in the appendix (cf. Appendix B), the fleet-based approach also showcased its advantages in recognizing actual issues. However, to quantitatively evaluate this improvement, the KG must first be converted into an operational tool with interfaces for the various user groups. Then, a long-term study in practical use can evaluate the advantages for specific queries and workflows and identify improvements for the data model.

In addition to the advantages demonstrated in this small example, some challenges and limitations should not go unmentioned. As a native graph database, Neo4j has its strengths when it comes to queries across multiple relations. For storing real-time data streams such as logs that are not to be further modified, document-based databases could be a faster and more suitable alternative (cf. Table 5.1 on page 109). The data model could remain valid, and the relevant data could be loaded into the graph database only when needed. Future investigations should consider blending these two approaches.

Another challenge is the availability of MBSE-based development artifacts. Currently, feature models, logical architecture, and TARA are not guaranteed to be available as digital models. However, as this lies outside the scope of cybersecurity engineering, the organizational level should promote model-based development. Depending on the modeling languages used, suitable integrations for the data model will also need to be developed in the future. For example, industrial practitioners would rather use AUTOSAR XML than the YAML files used in

the prototypical implementation. Finally, future works should re-examine the assumption that a central COD monitoring is available. Since research on ODD and its runtime verification is not yet very mature, however, the necessities in this field are yet to emerge. In the long term, even greater integration of safety and cybersecurity monitoring could be conceivable, for example, by transferring the graph concept to safety.

Immediate incident response

The final contribution of this thesis is immediate incident response from the VSOC, which the implemented example also partly demonstrated. Although the incident management application and response controller are only initial prototypes, they can already illustrate the support for real-time analysis and mitigation with a few steps. Showcasing the potential of immediate responses over the air, the developed CIM implements the log-level action enabling detailed incident investigation. The firewall's actions effectively contained the impact of the demonstrated security event by isolating the compromised vehicle.

For evaluation, the current implementation misses a mechanism to verify the effectiveness of automated responses, risking unresolved incidents. Additionally, the security of activating the response is limited. For example, future research could evaluate the integration of MFA³⁵, including exploring how multiple VSOCs around the globe could collaborate. A fundamental challenge remains the safety certification of adaptable security measures, as dynamic responses may conflict with the stringent, static requirements of safety-critical systems.

³⁵ Multi-factor authentication (cf. glossary, page 327)

12 Summary and future work

As a result of the digital revolution currently sweeping through the automotive industry, software-based functions and connectivity are making vehicles increasingly intelligent. At the same time, there is a growing risk of cyberattacks that jeopardize security, data protection, and even the functionality of entire fleets. Therefore, UN Regulation 155 provides manufacturers with regulatory grounds for development and operation, and the ISO/SAE 21434 standard defines a framework for cybersecurity throughout the life cycle.

This dissertation examines the current state of research on the cybersecurity of vehicles during operation and explores what is necessary for long-term maintenance of the vehicles' cybersecurity. Central tasks of a manufacturer after production, that should be covered procedurally and technologically, include monitoring, incident and vulnerability analysis, updating risk assessments, and incident response. At its core, long-term maintenance centers around establishing a central responsibility for the operational cybersecurity of a manufacturer's vehicle fleet, the VSOC. Accordingly, the state-of-the-art analysis considered available approaches from IT SOC and VSOC, incident response, and monitoring technologies. Although established IT security standards (e.g., NIST SP 800-61) and monitoring procedures (e.g., data models, network security monitoring) exist, they do not cover the specific challenges of vehicle cybersecurity. On the other hand, the automotive standard ISO/SAE 21434 shows gaps compared to the IT standards, for example, in incident handling and deriving monitoring requirements from risk assessments. Technically, existing automotive monitoring and incident response approaches insufficiently integrate in-vehicle and backend data, leaving

significant potential for structured data management and cybersecurity situation-awareness. These limitations underline the need for new, integrative technologies and processes for continuous vehicle cybersecurity operations.

12.1 Thesis contributions

What drives this work's contributions toward VSOCs is cybersecurity situation awareness, a term established by IT cybersecurity and cyber defense. Cybersecurity situation awareness emphasizes that a holistic understanding of the system's situation and tasks is essential to accomplish the necessary cybersecurity activities, including vulnerability analysis, developing updates, long-term improvement of risk analyses and countermeasures, and acute incident response. In this realm, three research questions were defined (cf. Section 1.5 on page 13). Overall, this dissertation provides six key contributions to address the research questions and improve the continuous cybersecurity of future vehicles. A prototypical implementation of a headlight assistant system that is exploited and monitored demonstrated the procedural and technological contributions.

Operations process

The first research question, which revolves around the phases and processes of cybersecurity engineering in operation, is answered by the concept of a new, independent cybersecurity operating process.

Contribution 1:

A cybersecurity operations process, that includes separate incident and vulnerability handling and a post-incident analysis phase, extending the current state-of-the-art methodology of ISO/SAE 21434.

The process is aligned with IT security terminology to streamline interaction with production and IT-related cybersecurity processes. It establishes continuity

through continuous improvement, including updates of risk assessments. The proposed process has been submitted to the standardization committee as a suggested enhancement to the new revision of the ISO/SAE 21434 standard.

Monitoring

The second research question is the definitive focus of this thesis. What data is applicable, and how it can be collected and pre-processed, is answered by one procedural and three technological contributions.

First of all, there is no universal answer to the question of what information should be collected, nor would it be sensible to provide one, as this would disregard the specific risks associated with a particular vehicle. In general, this thesis proposes the development of a holistic monitoring system that combines development artifacts, operational data (collected from the vehicle fleet), and data from external sources. Gathering relevant data from specific vehicles has to commence in the development phase by implementing proper monitoring measures. To this end, this thesis establishes a development methodology that determines suitable monitoring requirements for a particular architecture.

Contribution 2:

A structured method to define in-vehicle cybersecurity monitoring requirements based on a refined TARA that maps attack steps to threat information.

The implementation of the requirements in a specific vehicle provides the necessary sensors to monitor the in-vehicle network security. Combined with observing the vehicle's physical, safety-relevant context, this results in a tailored selection of relevant vehicle-internal data.

Contribution 3:

In-vehicle data collection for security-relevant data, combining network security monitoring with operational context information.

While the identification of critical, malicious activities requires enough suitable data to analyze, the collected information should not overwhelm the staff in the VSOC. Accordingly, this work proposes intelligent filtering in the vehicle, limiting the continuous stream of information to critical events but buffering the low-priority information for further investigation.

Contribution 4:

Configurable centralized buffering and filtering of information in the vehicle and secure fleet-wide collection, providing a suitable amount of in-vehicle information.

Ultimately, the question is also how the collected and selected data must be processed so that the operational process can utilize it. While an analyst could also rummage through a pile of tables and texts and eventually discover something critical, fast and targeted analysis and reactions require a more intelligent approach. To this end, this thesis proposes focusing on linking data sources and views in the form of a KG, which enables analysts to query for relationships between vehicles, configurations, contexts, alerts, historical incident handling actions, and threats.

Contribution 5:

A graph-based data model to combine and prepare data from in-vehicle monitoring, enriched by additional knowledge, including development artifacts, reports and CTI.

Incident response

The final research question revolves around the fastest possible response capability. The answer from the perspective of this work is not to rely solely on

the development of updates, but to integrate preventive response options into vehicles. Similar to control-over-the-air strategies for outsourcing functions from the vehicle to the cloud, the necessary measures, already planned and approved during development, are activated from the VSOC to respond immediately to incidents.

Contribution 6:

Introducing immediate response to incidents (and vulnerabilities) by pre-homologating containment, eradication and observation actions, leveraging the vehicle's service-oriented architecture.

12.2 Outlook

This work has contributed several technological and procedural elements to secure vehicles throughout their life cycle, combining holistic monitoring, continuous improvement, and immediate response into a coherent foundation. Advancing these contributions toward larger practical applications is a natural next step. Only long-term use within an automotive organization will prove the practicability of the proposed operations process in detail and reveal where adjustments are needed. The individual contributions, however, already point toward concrete research directions worth pursuing.

In practice, the success of the proposed monitoring requirements derivation methodology depends on correctly assessing the detection capabilities of sensors sourced from suppliers. The current methodology uses an unweighted mapping of attacker techniques to observables, and assesses sensors by whether they can in principle trace a given observable. Future work could substantially refine this by introducing the concept of *detection feasibility*: a characterization of how reliably a given sensor can observe a technique on a particular asset, parameterized analogously to the attack feasibility rating in ISO/SAE 21434, which could include temporal and contextual constraints on the detection side. Calibrating these parameters through simulation and real-world attack testing would

transform sensor selection and allocation from a binary coverage problem into a proper optimization problem, and could unlock the full potential of the proposed methodology at scale.

A particular strength of the proposed approaches is that they are designed for continuity. Attackers continuously evolve their techniques and tactics, and a CTI catalog should never be considered finished. The same holds for the KG data model. While the architecture and development process elements are in principle stable, the graph structure is inherently extensible, accommodating new cybersecurity knowledge and its interrelationships as the threat landscape evolves.

Regarding the proposed KG, its potential reaches considerably broader across the cybersecurity operations process than demonstrated in this work, from detection and triage to hypothesis generation and response selection. AI will likely play a key role in making this potential operational, supporting analysts who face growing data volumes and alert fatigue, or automating tasks entirely. Beyond operations, a unified KG foundation also offers a path to closing the loop toward development: ensuring the effectiveness of monitoring and prevention measures, triggering updates to risk analysis, and generating cybersecurity test scenarios from real operational knowledge. One could argue that the latter three are perhaps the most impactful directions, as they connect the operational reality back into the engineering processes that automotive cybersecurity regulation and standards demand. These are largely open research challenges, and a structured, semantically rich data model is a prerequisite for addressing them.

Perhaps the most ambitious direction opened by this work is the convergence of safety and cybersecurity monitoring. At both the procedural and technological levels, the structural analogy between HARA and TARA, the parallel runtime monitoring of conditions and requirement fulfillment in functional safety and SOTIF, and the mutual interference of safety and security mechanisms at runtime all point toward a unified development as well as observational layer. This thesis already embraces this thought using the current operational domain or ODD as physical context for cybersecurity monitoring. A joint monitoring system might detect anomalous, insecure or unsafe behavior without assuming a root cause,

then enable attribution across security incidents, functional faults, and SOTIF triggering conditions. This could potentially surface unknown unsafe scenarios, or challenge situation classifications previously assumed safe. Transferring the KG and its data model to functional safety and SOTIF monitoring, and connecting the corresponding views, could provide exactly such a unified observational layer across all three domains.

To sum up, the vehicle is perhaps one of the most demanding instances of a connected cyber-physical system for cybersecurity, being safety-critical, long-lived, distributed across millions of units, and increasingly software-defined. While markets and past certainties are changing dramatically today, and a profound transformation is taking place in automotive industrial organizations, from a research perspective, even 140 years after the construction of the first vehicle, there is no shortage of challenges and opportunities to address. With connectivity and driving automation continuing to advance, cybersecurity will remain an essential and evolving requirement. Uncovering individual vulnerabilities is crucial to address current problems, but securing the entire product and its operation over its full lifecycle is another deep research opportunity. In this respect, future research can draw on the contributions of this thesis, because what is certain is that no connected system is 100% secure.

A Further details on the outcomes of this work

A.1 Terminology of the resulting process

A relationship diagram of the terminology used in the operations process proposed this thesis is depicted in Figure A.2. The terminology of TARA defined in ISO/SAE 21434 is shown in Figure A.1.

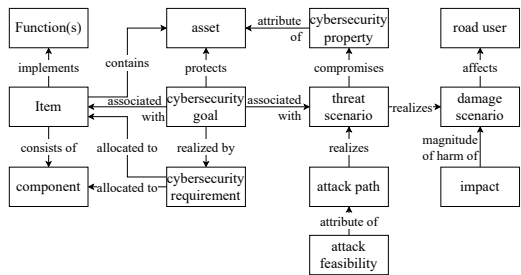


Figure A.1: Terms used in the TARA and their relationship (derived from and extended Fig. 3 in ISO/SAE 21434 [ISO21]).

The full terminology table for the developed process is depicted in tables A.1, A.2 and A.3.

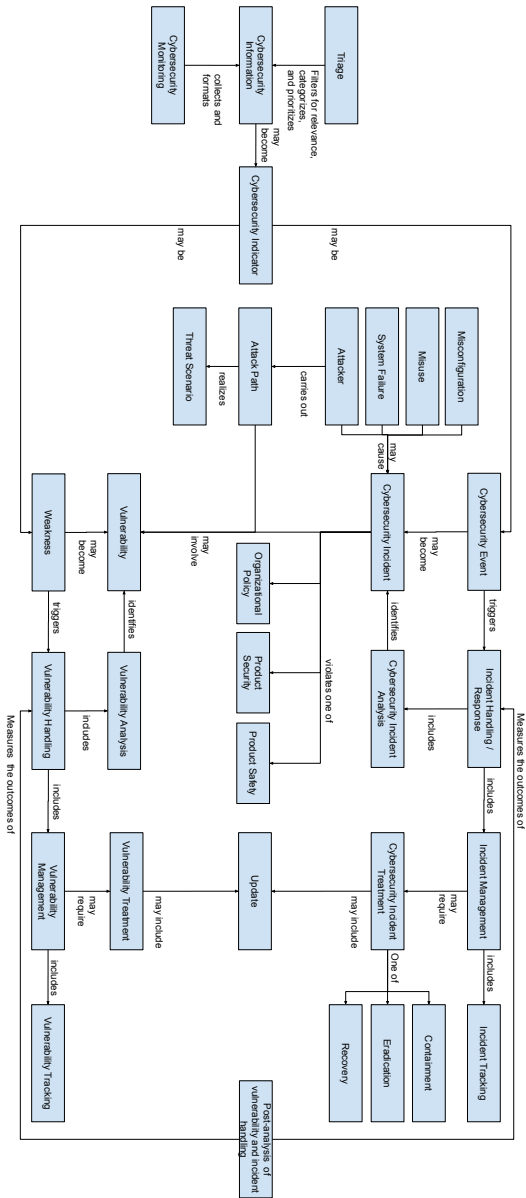


Figure A.2: Terminology relationships.

Table A.1: Terminology comparison of different standards and reports. UNR 155 and MITRE have no definitions for the terms listed in this table.

Term	ISO/SAE 21434	NIST SP 800-61	CMU/SEI-2004-TR-015	ISO 27035
Attack / Attack path	set of deliberate actions to realize a threat scenario	-	-	attempt to destroy, expose, alter, disable, steal or gain unauthorized access to or make unauthorized use of an asset
Cybersecurity event	cybersecurity information that is relevant for an item or component	any observable occurrence in a network or system	[Shi00]: an occurrence in a system that is relevant to the security of the system ¹	information security event: occurrence indicating a possible breach of information security or failure of controls
Cybersecurity incident	situation in the field that can involve vulnerability exploitation	A violation or imminent threat of violation of computer security policies, acceptable use policies, or standard security policies	[Shi00]: a security event that involves a security violation ² . [GB98]: computer security incident: any adverse event which compromises some aspect of computer or network security	information security incident: one or multiple related identified information security events that can harm an organization's assets or compromise its operations
Cybersecurity incident handling	-	The mitigation of violations of security policies and recommended practices (=Incident response)	the processes used for handling an incident; in this text, the term includes the processes for detecting, reporting, triaging, analyzing, and responding to computer security incidents.	actions of detecting, reporting, assessing, responding to, dealing with, and learning from information security incidents

¹ includes both events that are incidents and those that are not

² In other words, a security-relevant system event in which the system's security policy is disobeyed or otherwise breached

Terminology comparison of different standards and reports (ctd.)

Term	ISO/SAE 21434	NIST SP 800-61	CMU/SEI-2004-TR-015	ISO 27035
Cybersecurity incident management	-	-	the processes for controlling or administering tasks associated with computer security incidents; in this text, the term implies management of a computer security incident, and includes all of the Detect, Triage, and Respond processes as well as the Prepare (improve, sustain) processes and the Protect processes outlined in this report.	information security incident management: exercise of a consistent and effective approach to the handling of information security incidents
Cybersecurity incident response	-	The mitigation of violations of security policies and recommended practices (=Incident handling)	an answer given or action taken by people designated to react to an incident. It is the process that encompasses the planning, coordination, and execution of any appropriate mitigation and recovery strategies and actions.	actions taken to mitigate or resolve an information security incident, including those taken to protect and restore the normal operational conditions of an information system and the information stored in it
Cybersecurity information	information with regard to cybersecurity for which relevance is not yet determined	-	-	-
Indicator	-	A sign that an incident may have occurred or may be currently occurring	-	-
Threat scenario	potential cause of compromise of cybersecurity properties of one or more assets in order to realize a damage scenario	-	-	Threat: potential cause of an unwanted incident, which may result in harm to a system or organization

Terminology comparison of different standards and reports (ctd.)

Term	ISO/SAE 21434	NIST SP 800-61	CMU/SEL-2004-TR-015	ISO 27035
Triage	analysis to determine the relevance of cybersecurity information to an item or component	-	the process of receiving, initial sorting, and prioritizing of information to facilitate its appropriate handling	-
Update	Not a definition, but a related description: "Updates are changes made to an item or component during post-development and can include additional information, e.g. technical specifications, integration manuals, user manuals."	-	-	-
Vulnerability	weakness that can be exploited as part of an attack path	A weakness in a system application or network that is subject to exploitation or misuse	[GB98]: a characteristic of a piece of technology which can be exploited to perpetrate a security incident. [Wes+03]: the existence of a software weakness, such as a design or implementation error, that can lead to an unexpected, undesirable event compromising the security of a system, application or protocol	weakness of an asset or control that can be exploited by a threat
Vulnerability analysis	Not a definition, but a description: examines weaknesses and assesses if a particular weakness can be exploited.	-	Vulnerability assessment: an act or procedure intended to evaluate or identify the existence of known vulnerabilities (in a computer system or network)	-

Terminology comparison of different standards and reports (ctd.)

Term	ISO/SAE 21434	NIST SP 800-61	CMU/SEI-2004-TR-015	ISO 27035
Vulnerability management	Not a definition, but a description: tracks and oversees the treatment of identified vulnerabilities in items and components until their end of cybersecurity support.	-	-	-
Weakness	defect or characteristic that can lead to undesirable behavior	-	-	-

Table A.2: Terminology comparison of different standards and reports.

Term	UNR 155 [IEC21a]	UNR 156 [IEC21b]	MITRE [The], Common Weakness Exposure Project
Threat	"threat" means a potential cause of an unwanted incident, which may result in harm to a system, organization or individual	-	-
Update	-	"software update" means a package used to upgrade software to a new version including a change of the configuration parameters	-
Vulnerability	weakness of an asset or mitigation that can be exploited by one or more threats	-	an occurrence of a weakness (or multiple weaknesses) within a product, in which the weakness can be used by a party to cause the product to modify or access unintended data, interrupt proper execution, or perform incorrect actions that were not specifically granted to the party who uses the weakness. a type of mistake that, in proper conditions, could contribute to the introduction of vulnerabilities within that product. This term applies to mistakes regardless of whether they occur in implementation, design, or other phases of a product life-cycle.
Weakness	-	-	-

Table A.3: Chosen terminology definitions for this thesis (cf. tables A.1 and A.2)

Term	Chosen definition*	status**
Attack / Attack path	set of deliberate actions to realize a threat scenario	S
Cybersecurity event	an occurrence in an item/component that is relevant to the cybersecurity of the system	D
Cybersecurity incident	a cybersecurity event that involves a violation of product cybersecurity, product safety or organizational cybersecurity policy	D
Cybersecurity incident analysis	the process of identifying and assessing cybersecurity incidents from identified cybersecurity events	U
Cybersecurity incident handling	the process of analysing and managing cybersecurity incidents	U
Cybersecurity incident management	the process of tracking and treating cybersecurity incidents, including actions for containment, eradication or recovery	U
Cybersecurity incident response	the process of analysing and managing cybersecurity incidents (NOTE: same as incident handling)	P
Cybersecurity information	information with regard to cybersecurity for which relevance is not yet determined	S
Cybersecurity monitoring	the process of collecting, pre-processing and formatting cybersecurity information for subsequent processing	P
Indicator	information with regard to cybersecurity that is relevant for an item or component	U
Post-analysis of vulnerability and incident handling	the process of analysing and measuring incident and vulnerability handling outcomes, including technical root cause analysis and efficiency measurements	U
TARA management process	the process of brokering access to and coordinating work on TARA artifacts required for an item, including the coordination of distributed development activities and the definition of appropriate abstraction levels	U
Threat scenario	potential cause of compromise of cybersecurity properties of one or more assets in order to realize a damage scenario	S
Triage	the process of determining relevance of cybersecurity information and prioritization of resulting indicators to facilitate their appropriate handling	D
Update	a package used to upgrade software to a new version including a change of the configuration parameters	P
Vulnerability	weakness that can be exploited as part of an attack path	S
Vulnerability analysis	the process of identifying and assessing vulnerabilities from identified weaknesses, which includes performing the appropriate steps of threat analysis and risk assessment	P

* Could be considered for inclusion in next ISO/SAE 21434 revision

** ISO/SAE 21434 status legend: D = Defined differently; P = Process/concept exists without definition; S = Same definition; U = Undefined

Chosen terminology definitions for this thesis (ctd.)

Term	Chosen definition*	status**
Vulnerability handling	the process of performing vulnerability analysis and vulnerability management	U
Vulnerability management	the process of tracking and treating cybersecurity vulnerabilities	P
Weakness	defect or characteristic that can lead to undesirable behavior	S

* Could be considered for inclusion in next ISO/SAE 21434 revision

** ISO/SAE 21434 status legend: D = Defined differently; P = Process/concept exists without definition; S = Same definition; U = Undefined

A.2 The ASCOT threat catalog

The ASCOT matrix or threat catalog involves several tables listing techniques, observables and cybersecurity monitoring sensors relevant to automotive systems. Applying the methodology for identifying monitoring requirements requires usage of the ASCOT catalog. Supporting this task, a web-based tool was developed, derived from the open-source source platform of MITRE ATT&CK®³. The new automotive TTP catalog denoted as *ASCOT matrix* complements the existing Enterprise, Mobile, and industrial control systems perspectives. The tool enables to search and filter the content, including viewing the observables (cf. Figure A.4) per technique besides providing an overview of the techniques per tactic (cf. Figure A.3). The tool generates its representations automatically from STIX files⁴, containing the techniques, tactics, and observables. Moreover, the KG's threat information view is built from the STIX files. Updating the tool new knowledge about attackers and their methods thus only requires to adding or change the respective STIX objects.

³ <https://github.com/mitre-attack/attack-website>

⁴ STIX objects stored in a JSON file

A Further details on the outcomes of this work

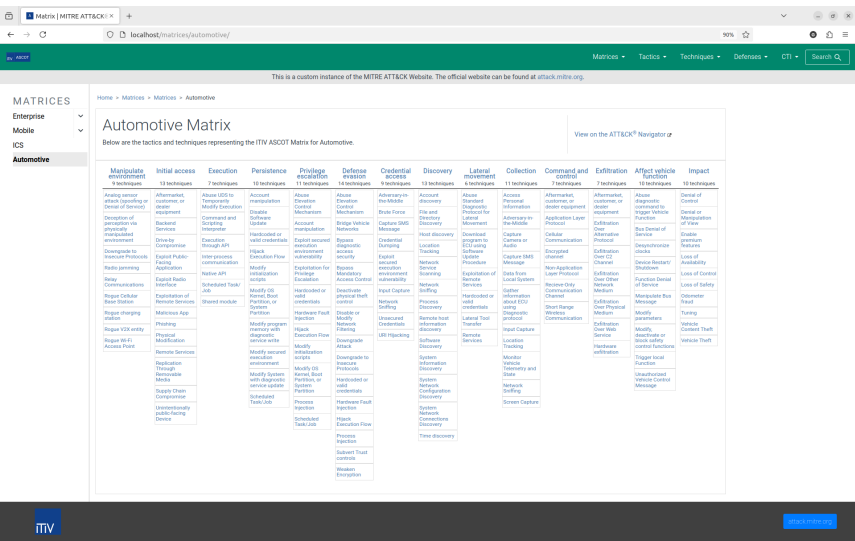


Figure A.3: Overview of the tactics and techniques in the ASCOT threat catalog.

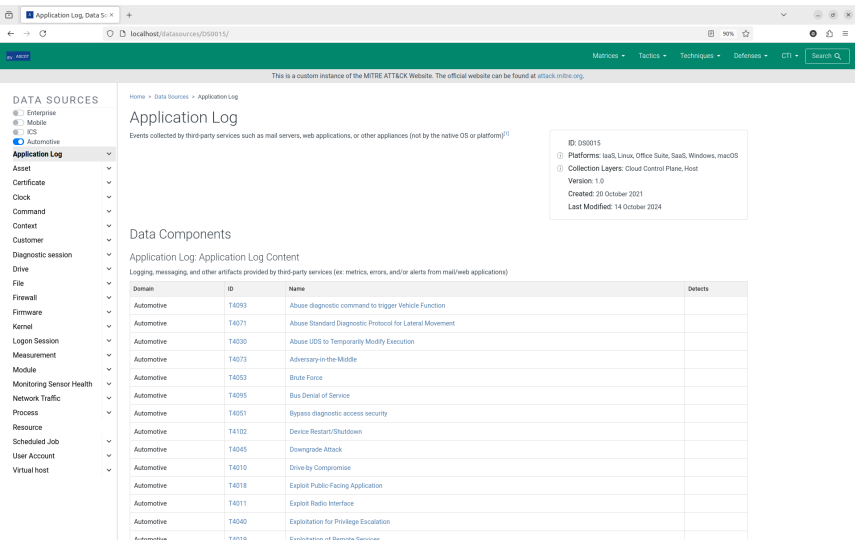


Figure A.4: List of observable information that were mapped to techniques. Note, that MITRE denotes the observables as "Data Sources".

Table A.4: Techniques and tactics

|--|

A Further details on the outcomes of this work

Techniques and tactics (ctd.)

ID	Technique	ME	IA	Ex	Ps	PE	DE	CA	Di	LM	Co	CC	Ef	AF	Im	Observables	ATT&CK® reference
13	Malicious App		x													3, 11, 14	T0819 T0866 T0867 T0886
14	Phishing		x													1, 4, 11, 20	
15	Physical Modification		x													1, 3, 7, 21	
16	Supply Chain Compromise		x									x				3, 11	
17	Aftermarket, customer, or dealer equipment		x										x			1, 3, 5, 21	
18	Exploit Public-Facing Application		x													1, 6	
19	Exploitation of Remote Services		x													1, 6, 21	
20	Lateral Tool Transfer		x							x						1, 11, 13	
21	Remote Services		x							x						1, 4, 5, 13, 14, 16, 21	
22	Backend Services		x													1, 4, 20, 21	
23	Unintentionally public-facing Device		x													1, 4	
24	Command and Scripting Interpreter			x												13, 14	
25	Native API			x												14	
26	Execution through API			x												1, 14	
27	Scheduled Task/job			x									x			11, 12, 13, 14, 22	
28	Shared module			x												14, 16	
29	Inter-process communication			x												14, 16	
30	Abuse UDS to temporarily modify execution			x												1, 5, 6	
31	Modify system with diagnostic service update				x											1, 3, 5, 6	
32	Modify program memory with diagnostic service write				x											1, 5, 6	
																	T0889

Techniques and tactics (ctd.)

		Im	AF	Ef	CC	Co	LM	Di	CA	DE	PE	Ps	Ex	IA	ME	ID	Technique		Tactics	Observables	ATT&CK® reference
33	Disable Software Update											x					33	Disable Software Update		13, 14	T1542,
34	Modify OS Kernel, Boot Partition, or System Partition										x	x					34	Modify OS Kernel, Boot Partition, or System Partition		3, 14, 15	T1547/006
35	Modify initialization scripts											x					35	Modify initialization scripts		11, 13, 14	T1037/004
36	Modify secured execution environment											x					36	Modify secured execution environment	14		T1399
37	Account manipulation										x	x					37	Account manipulation	11, 13, 14, 20		T1098, T1136
38	Hardcoded or valid credentials									x	x	x					38	Hardcoded or valid credentials	1, 4, 20		T0891, T0859
39	Abuse Elevation Control Mechanism										x	x					39	Abuse Elevation Control Mechanism	11, 13, 14		T1548
40	Exploitation for Privilege Escalation										x						40	Exploitation for Privilege Escalation	6, 14, 21		T1068, T0890
41	Exploit secured execution environment vulnerability								x		x						41	Exploit secured execution environment vulnerability	14		
42	Hardware Fault Injection									x	x						42	Hardware Fault Injection	6		T1055/008
43	Process Injection									x	x						43	Process Injection	11, 14, 16		T1574
44	Hijack Execution Flow									x	x	x					44	Hijack Execution Flow	11, 13, 14, 16		T1562/010,
45	Downgrade Attack											x					45	Downgrade Attack	1, 3, 6, 13, 14		T1601
46	Weaken Encryption											x					46	Weaken Encryption	11, 14		T1600
47	Deactivate physical theft control										x	x					47	Deactivate physical theft control	1		
48	Subvert Trust controls										x						48	Subvert Trust controls	10, 11, 13		T1553
49	Disable or Modify Network Filtering										x	x					49	Disable or Modify Network Filtering	13, 14, 18		T1562/004
50	Bridge Vehicle Networks																50	Bridge Vehicle Networks	1		
51	Bypass diagnostic access security										x						51	Bypass diagnostic access security	1, 5, 6, 21		

Techniques and tactics (ctd.)

ID	Technique	Tactics													Observables	ATT&CK® reference
		ME	IA	Ex	Ps	PE	DE	CA	Di	LM	Co	CC	Ef	AF	Im	
52	Bypass Mandatory Access Control															T1562/001
53	Brute Force						x	x								T1110
54	Input Capture							x			x					T1056
55	Network Sniffing							x	x		x					T1040
56	Credential Dumping							x								T1003
57	Credential Dumping							x								
58	Unsecured Credentials							x								
59	URI Hijacking							x								T1635/001
60	Account discovery								x							
61	Host discovery								x							
62	File and Directory Discovery								x							
63	Location Tracking								x							
64	Time discovery								x		x					
65	Network Service Scanning								x							
66	Process Discovery								x							
67	Software Discovery								x							T1046
68	System Information Discovery								x							
69	Remote host information discovery								x							
70	System Network Configuration Discovery								x							
71	System Network Connections Discovery								x							
72	Abuse Standard Diagnostic Protocol for Lateral Movement															

Techniques and tactics (ctd.)

ID	Technique	ME	IA	Ex	Ps	PE	DE	CA	Di	LM	Co	CC	Ef	AF	Im	Observables	ATT&CK® reference
		Tactics															
72	Download program to ECU using Software Up- date Procedure									x						1, 3, 20	T1557, T0830
73	Adversary-in-the-Middle							x			x					1, 6, 9, 10	
74	Gather information about ECU using Diagnostic protocol										x					1, 5	
75	Access Personal Information										x					11, 13	
76	Monitor Vehicle Telemetry and State										x					1, 6	
77	Capture Camera or Audio										x					1, 11, 13	
78	Capture SMS Message							x			x					11, 13	
79	Data from Local System										x					11, 13	
80	Screen Capture										x					13	
81	Application Layer Protocol											x				1	
82	Non-Application Layer Protocol											x				1	
83	Cellular Communication											x				1, 21	
84	Recieve-Only Communication Channel											x				1, 6	
85	Short Range Wireless Communication											x				1, 21	
86	Encrypted channel											x				1, 10	
87	Exfiltration Over C2 Channel											x				1, 11, 13	
88	Exfiltration Over Other Network Medium												x			1, 11, 13, 21	
89	Exfiltration Over Physical Medium												x			11, 13, 14, 19	
90	Exfiltration Over Alternative Protocol												x			1, 11, 13	
91	Exfiltration Over Web Service												x			1, 11, 13	

A Further details on the outcomes of this work

Techniques and tactics (ctd.)

		Im	AF	Ef	CC	Co	LM	Di	CA	DE	PE	Ps	Ex	IA	ME			ATT&CK® reference	
ID	Technique	Tactics													Observables				
92	Hardware exfiltration																21		
93	Abuse diagnostic command to trigger Vehicle Function		x														1, 5, 6		
94	Modify parameters		x														1, 3, 5, 6		
95	Bus Denial of Service		x														1, 6, 21		
96	Function Denial of Service		x														1, 6, 21		
97	Trigger local Function		x														1, 6, 13, 14		
98	Manipulate Bus Message		x														1, 6, 7		
99	Unauthorized Vehicle Control Message		x														1, 6, 7		
100	Modify, deactivate or block safety control functions		x														1, 6, 7, 13, 14, 18		
101	Desynchronize clocks		x														1, 2	T0816	
102	Device Restart/Shutdown		x														1, 6, 13, 21		
103	Loss of Availability															x	21		
104	Loss of Safety																x	7, 21	T0880
105	Denial or Manipulation of View																x	23	T0815
106	Loss of Control																x	21	T0827
107	Denial of Control																x	1, 21	
108	Enable premium features																x	1, 3	
109	Tuning																x	7	
110	Odometer fraud																x	7	
111	Vehicle Theft																x	21, 23	

Techniques and tactics (ctd.)

ID	Technique	Tactics	Observables	ATT&CK® reference
112	Vehicle Content Theft	ME		
		IA		
		Ex		
		Ps		
		PE		
		DE		
		CA		
		Di		
		LM		
		Co		
		CC		
		Ef		
		AF		
		Im	x	23
113	Loss of Privacy		x	1, 11, 13

Table A.5: Observables and reference to corresponding data sources in MITRE ATT&CK®.

Identifier o_t	Observable	Description	ATT&CK® reference
1	Network traffic	Data transmitted across a network (ex: Web, DNS, Mail, File, etc.), that is either summarized (ex: Netflow), and/or captured as raw data in an analyzable format (ex: PCAP)	DS0029
2	Clock	Software and hardware to provide current system time on a host. In real-time systems, different hosts may require synchronized clocks for correct timing behavior, such as sequences of calculation.	
3	Asset	Set of devices (hardware; electronic control units, sensors, performance computers and others capable of running software) found within the network, along with their current software (Computer program, including firmware for low-level hardware control such as BIOS and bootloader, and application software) and configurations	DS0001 / DS0039
4	Logon session	Logon occurring on a system or resource (local, domain, or cloud) to which a user/device is gaining access after successful authentication and authorization	DS0028
5	Diagnostic session	Logon occurring on a system to perform diagnostic activities which a device is gaining access after successful authentication and authorization	DS0028
6	Application log	Events collected by third-party services such as mail servers, web applications, or other appliances (not by the native OS or platform)	DS0015
7	Measurement	An individual piece of information about the operation of the system, such as signals (including historical and real-time information), interface fields, parameters, and alarms or events produced to indicate unusual or concerning activity within the operational process	DS0040
8	Resource	Hardware component or system capability that provides computing power and storage for processes.	
9	Domain Name	Information obtained (commonly through registration or activity logs) regarding one or more IP addresses registered with human readable names (ex: mitre.org)	DS0038
10	Certificate	A digital document, which highlights information such as the owner's identity, used to instill trust in public keys used while encrypting network communications	DS0037
11	File	A computer resource object, managed by the I/O system, for storing data (such as images, text, videos, computer programs, or any wide variety of other media).	DS0022
12	Scheduled job	Automated tasks that can be executed at a specific time or on a recurring schedule running in the background, including system background tasks (ex: Cron , /var/log/daemon)	DS0003 / DS0019
13	Command	A directive given to a computer program, acting as an interpreter of some kind, in order to perform a specific task	DS0017

Observables and reference to corresponding data sources in MITRE ATT&CK® (ctd.)

Identifier <i>o_I</i>	Observable	Description	ATT&CK® reference
14	Process	Instances of computer programs that are being executed by at least one thread, including execution of scripts. Processes have memory space for process executables, loaded modules (DLLs or shared libraries), and allocated memory regions containing everything from user input to application-specific data structure.	DS0009 / DS0012
15	Kernel module	A computer program, at the core of a computer OS, that resides in memory and facilitates interactions between hardware and software components	DS0008
16	Module	Executable files consisting of one or more shared classes and interfaces, such as portable executable (PE) format binaries/dynamic link libraries (DLL), executable and linkable format (ELF) binaries/shared libraries, and Mach-O format binaries/shared libraries	DS0011
17	Sensor Health	Information from information management and monitoring components about status, errors, or other notable functional activity	DS0013
18	Firewall	A network security system, running locally on a network element, that monitors and controls incoming/outgoing network traffic based on predefined rules	DS0018
19	Drive	A non-volatile data storage device (hard drive, floppy disk, USB flash drive) with at least one formatted partition, typically mounted to the file system and/or assigned a drive letter	DS0016
20	User Account	A profile representing a user, device, service, or application used to authenticate and access resources	DS0002
21	Context	Abstracted information describing the operational situation of the system, including environment, status, time, and own behavior.	
22	Virtual host	A virtual host is a container or virtual machine, providing an enclosed environment of dependencies and limited hardware separation	
23	Customer	Customer detecting uncommon or dangerous behavior	

Table A.6: Cybersecurity monitoring sensors, domain and common potential observables.

Identifier s_i	Sensor	Domain	Observables
1	NIDS	Network	1, 3
2	Firewall	Network	1, 18
3	Packet capture	Network	1
4	Flow collector	Network	1, 3
5	SIDS	Network	7
6	Telemetry	Network	7, 21
7	OS	Host	4, 12, 13, 15, 18, 19, 20
8	HIDS	Host	1, 3, 4, 9, 11, 12, 13, 14, 15, 16, 18, 19, 20, 22
9	Access control	Host	1, 3, 4, 5, 11
10	Network stack	Host	1, 2, 3, 10, 21
11	Certificate management	Host	10
12	Service	Service	6, 21
13	Update management	System	3, 5, 20
14	Execution management	System	3, 8, 14, 22
15	Network management	System	1, 3, 9, 21
16	Cybersecurity information management	System	17

A.3 The ASCOT data model

Figure A.5 shows the combination of all layers. The classes of the data model are defined in Table A.7.

Table A.7: Overview of classes across views of the ASCOT data model.

View	Class	Description
	Vehicle	Specific produced vehicle, defined by VIN
Development	Configuration	Selection of features and system revision to derive a specific product (vehicle) (cf. definition in [Ana+20])
	Feature	Specific variant and revision (version) of a feature
	System revision	All feature revisions of the product line at a certain point in time
	System generation	Unchangeable type and generation of a vehicle
	Component	Hardware and/or software realizing a feature in a particular version, may consist of further sub-components
	Capability	Functionality provided or required by a Component

Table A.7: Overview of classes across views of the ASCOT data model (ctd.)

View	Class	Description
Development (ctd.)	Data	Interface type or particular signal of a Capability or configuration data of a Component
	Threat scenario	Compromises cybersecurity property and realizes a damage scenario, has a risk value
	Item	"Component or set of components realizing a function at vehicle level" [ISO21]
	Attack path	Actions to realize a threat scenario, has feasibility rating
	Attack step	Individual actions of an attack path including step realizing threat scenario
	Damage scenario	Adverse consequences, has impact rating
	Cybersecurity control	Component that modifies (reduces) the risk of a threat scenario
Vehicle shadow	Instance	Implemented (instantiated) component in produced vehicle
	Host	Instance of component, any computing unit that can execute software, either virtualized or physical, including ECUs
	Host cluster	Set of interconnected hosts that share and combine resources for the execution of software
	Host cluster endpoint	Specific host in a cluster that accepts control commands
	Sensor / Actuator	Hardware instance that is not providing computation resources
	Software	Component instance that provides functionality to the vehicle or the users
	Container	Software (instance) involving OS-level virtualization to isolate and bundle an application with its software dependencies
	Network service	Software (instance) capable of communicating over the network, usually involving ports/IDs and protocols
	Network element	Hardware: Host, Sensor/Actuator, gateways, switches, bus systems, telematic units
	Context	Base class for all contexts
	Scenery	Non-movable elements of the operating environment
	Dynamic elements	Movable objects of the operating environment
	Environmental conditions	Includes weather, atmospheric conditions, and connectivity
	Status	Status of a component's Instance or the Vehicle, including activation status, hardware shutdown, error conditions and resource consumption
	User status	Status of the driver and passengers

Table A.7: Overview of classes across views of the ASCOT data model (ctd.)

View	Class	Description
Vehicle shadow (ctd.)	Time	Local time of the Vehicle, including working hours
	Cybersecurity information	Information for which relevance is not yet determined, including observables (cf. Table A.5) and reports.
	Alert	Notification of a cybersecurity control about certain cybersecurity information that is flagged as erroneous or unauthorized, including its blocking (e.g., by an IDS)
	Network traffic	Packets, signals, or network flow
	Observation point	Network element that provides data for a cybersecurity control
	Network address	Identifier in the network (e.g., IP address, CAN identifier)
	Subnet	VLAN or IP subnet
Incident and Vulnerability Handling	Domain name	Name assigned to a server, resolves to an IP address
	Indicator	Information relevant for an item or Component based on cybersecurity information
	Report	Document with technical findings, especially assessments
	Security event	Indicator of an occurrence relevant for a Component
	Incident	Security event proving active violation of vehicle security, safety, or company security policy
	Weakness	Indicator of a defect or characteristic that can lead to undesirable behavior
	Vulnerability	Weakness that can be exploited
Threat information	Course of action	Response to Incidents and Vulnerabilities, aiming at containment, eradication, or recovery
	Threat catalog	External or internal provider of strategic information about threats, defined as tactics
	Tactic	High-level behavior of the attacker
	Technique	Technological approach and method used to carry out an attack
	Threat actor	Person, group, or nation-state actor using tools, malware, and techniques
	Tool	Legitimate software or device that can be used by threat actors to perform attacks
	Malware	Malicious code or software inserted into a system, usually covertly
	CVE	Entry in the Common Vulnerabilities and Exposures database

A.4 Response actions

Table A.8: Response actions and potential actuators.

Identifier	Goal	Action	Actuator
1	Containment	Disable public internet connection (any interface)	Firewall
2	Containment	Revoke vehicle's certificate(s)	Certificate management
3	Containment	Disable OEM server connection (features)	Firewall
4	Containment	Disable OEM server connection (Updates)	Firewall
5	Containment	Reduce traffic to safety-critical network segments	Access control, firewall
6	Containment	Reallocate service to isolated platform	Execution management
7	Containment	Shutdown ECU	Network management
8	Containment	Stop service	Execution management
9	Containment	Authenticate network segment	Network stack
10	Containment	Authenticate and encrypt network segment	Network stack
11	Containment	Full shutdown	Services
12	Containment	Trigger limp mode	Services, Execution management
13	Containment	Disable charging interface	Access control, firewall
14	Containment	Disable V2V / V2I interface	Access control, firewall
15	Containment	Disable WiFi interface	Access control, firewall
16	Containment	Disable cellular interface	Access control, firewall
17	Containment	Disable diagnostic connection (local)	Access control, firewall
18	Containment	Disable diagnostic connection (remote)	Access control, firewall
19	Eradication	Physical destruction and change in workshop	-
20	Eradication	Uninstall malicious application	Update management
21	Eradication	Erase firmware with diagnostic write	Update management
22	Eradication	Re-download current software version	Update management
23	Eradication	Rollback to current firmware version by A/B swap	Update management
24	Eradication	Restart ECU	Network management
25	Eradication	Restart service	Execution management
26	Observation	Change filter severity	Information management
27	Observation	Increase service log level	Services
28	Observation	Reduce threshold of anomaly-based sensors	NIDS, HIDS
29	Observation	Activate additional rule sets	NIDS, HIDS, Firewall, Access control
30	Recovery	Roll out new certificate(s)	Certificate management
31	Recovery	harden against re-occurrence	Update management (configuration update of cybersecurity measures)
32	Recovery	Apply updates	Update management

B Exemplary application of the procedure to specify monitoring requirements

To illustrate the proposed method, a recent real-world example is analyzed below. In January 2025, a vulnerability was discovered at the vehicle manufacturer Subaru that allowed any vehicle to be unlocked and started remotely, as well as enabling the retrieval of operational data such as location over an extended period of time [Cur25]. This would have made vehicle theft just as possible as the targeted tracking of vehicle movements.

The aim of the following analysis is to evaluate whether the proposed development methodology and holistic monitoring could have detected and prevented such an attack before customers suffered harm on a larger scale. To this end, the network topology was first reconstructed using online workshop manuals [Sub]. Based on this, a DFD for the keyless entry system was then designed (cf. Figure B.1). It can be assumed that Subaru performed or should have performed a TARA for this customer feature, and that the system is therefore suitable as an item for analysis.

Building on the DFD, an attack tree was developed that illustrates some of the threat scenarios and damage scenarios (without claiming to be exhaustive), and outlines attack paths (A–H) for carrying out these scenarios (cf. Figure B.2). The presumably best-known attacks on keyless access systems are replay and relay attacks, which have been classified under the threat scenario "Attacker spoofs legitimate authentication." In a replay attack (G), the authentication signals and messages transmitted by the physical key are jammed and recorded in parallel, to be subsequently replayed at an unguarded moment to unlock and start the vehicle.

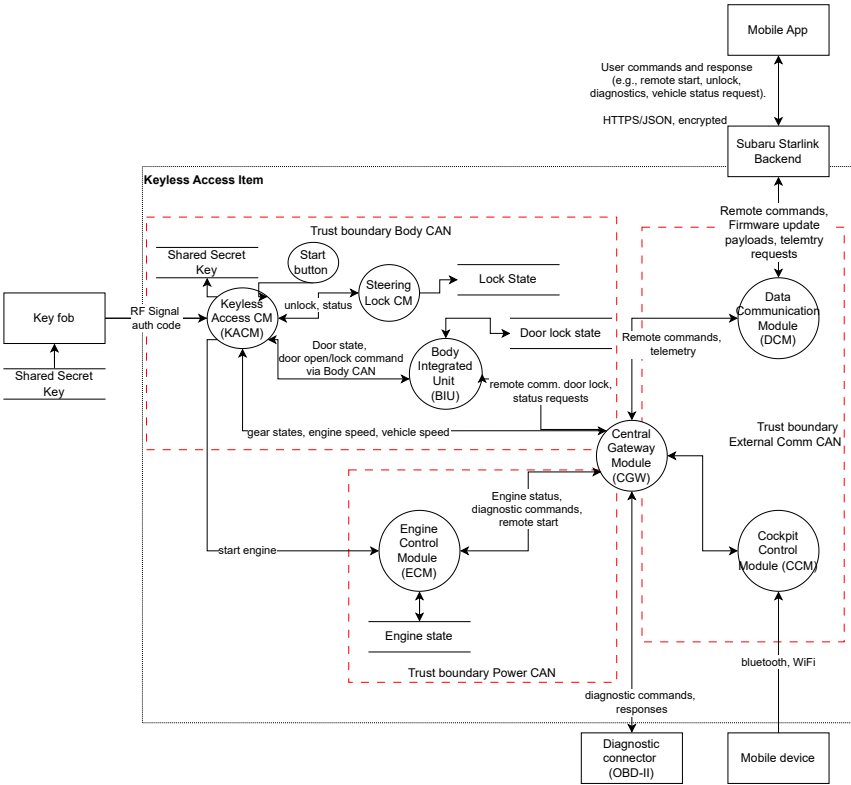


Figure B.1: Data flow diagram of the keyless access item.

The relay attack (H) bridges a physical distance between the key and the vehicle by amplifying and relaying signals to reach the same goal.

The TARA illustrates not only attacks via the physical key but also, in particular, attack vectors via Wi-Fi, Bluetooth, and the vehicle’s cellular interface (marked steps 1–12). This also includes various attack paths via the backend (paths D–F), such as malicious apps and phishing attacks on end users (see attack step 10 in Figure B.2). In practice, attack steps such as "Backend is compromised, allowing unauthorized command issuance" should be further detailed.

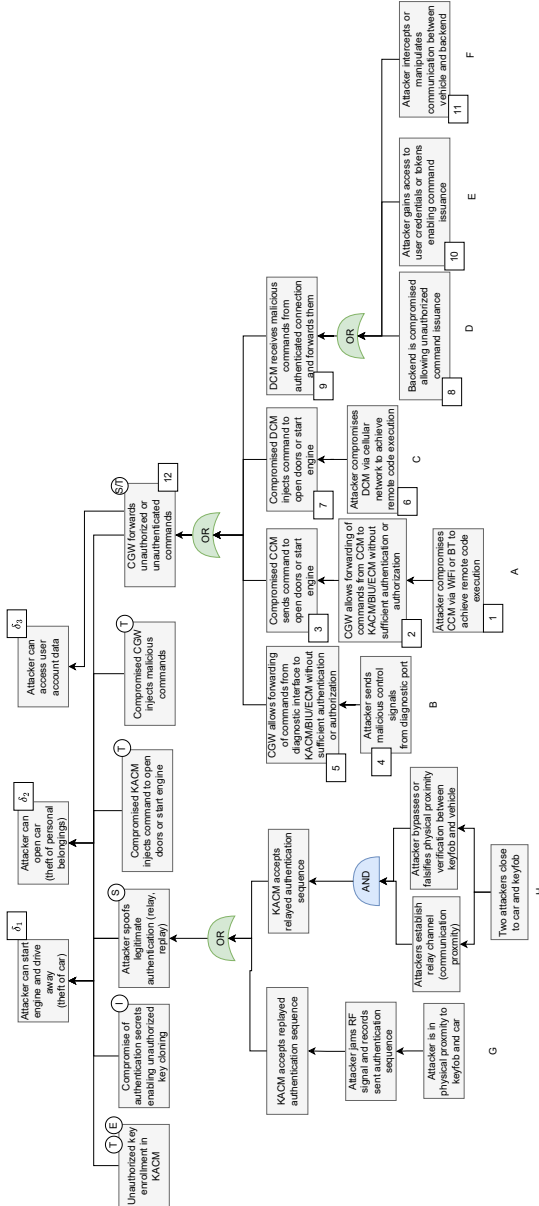


Figure B.2: Attack tree for the keyless access item.

Table B.1: Attack steps in the Keyless access TARA. ID references techniques (cf. Table A.4).

Step	Path	Technique	ID	Tactic	Monitoring goal
1	A	Exploit Radio Interface	11	Initial access	x
2	A	Disable or Modify Network Filtering	49	Defense evasion	
3	A	Unauthorized Vehicle Control Message	99	Affect vehicle function	
4	B	Aftermarket, Customer, or Dealer Equipment	17	Initial access	x
5	B	Disable or Modify Network Filtering	49	Defense evasion	
6	C	Exploit Radio Interface	11	Initial access	x
7	C	Manipulate Bus Message	98	Affect vehicle function	
8	D	Backend Services	22	Initial access	x
9	D, E, F	Hardcoded or valid credentials, Execution through API	38, 26	Privilege escalation, Execution	x
10	E	Malicious App, Phishing	13, 14	Initial access, Credential acces	x
11	F	Rogue Cellular Base Station, Adversary-in-the-Middle	7, 73	Manipulate environment, Credential access	x
12	A-F	Bridge Vehicle Networks	50	Lateral Movement	x
δ_1	A-H	Vehicle theft	111	Impact	x
δ_2	A-H	Vehicle content theft	112	Impact	x
δ_3	A-H	Loss of privacy	113	Impact	x

All attack steps were classified according to the attack techniques (cf. Table B.1). The paths were evaluated using the scoring scheme proposed in HEAVENS 2.0 [LAO21] (cf. Section). To estimate the impact, the logarithmic scale proposed in HEAVENS 2.0 was used for the individual evaluation categories, and an equal weighting of the categories¹ was applied, so that the damage scenarios yielded the impact values $\delta_1 : \frac{1+10+100+0}{400} = 0.2775$ (major), $\delta_2 : \frac{0+1+0+0}{400} = 0.0025$ (negligible), and $\delta_3 : \frac{0+0+0+10}{400} = 0.025$ (moderate). For attack feasibility, particular consideration was given to the simplicity of the equipment and the scalability of backend attacks. The results show that backend paths D and E should be classified as high risk (cf. risk level 5 in Table B.2) and that the attack steps should be monitored accordingly (cf. monitoring goals in Table B.1). However, the other paths are also of moderate with risk values of 3 or 4, meaning that the threat scenarios and leaf steps should be monitored as well.

¹ Safety-Financial-Operational-Privacy

Table B.2: Attack feasibility (AF) rating and risk value per attack path for keyless access item. Ratings shorthand with: (Ex)pertise, (K)nowledge of Item, (W)indow of Opportunity, (Eq)uipment.

Path	Description	Ex	K	W	Eq	AF value	AF rating	Impact rating	Risk value
A	Local compromise of CCM, transmission via CGW, triggering of commands in KACM, BIU, ECM	1	1	2	1	0.42	Low	Major	3
B	Attack via OBD and compromise of CGW	1	1	1	3	0.50	Low	Major	3
C	Compromised DCM via cellular interface, transmission via CGW, triggering of commands in KACM, BIU, ECM	1	2	3	1	0.58	Low	Major	3
D	Compromised backend, send commands via API	1	3	3	3	0.83	High	Major	5
E	Compromised customer's mobile phone or steal credentials, send commands via backend and API	2	3	3	2	0.83	High	Major	5
F	MITM on cellular connection to send commands via API	1	2	2	2	0.58	Low	Major	3
G	Replay attack	1	1	2	2	0.50	Low	Major	3
H	Relay attack	1	3	2	2	0.67	Medium	Major	4

This example does not consider specific sensor products, but rather assumes a generic applicability of sensors for monitoring certain observables (cf. Table A.6). Not all of the observables mentioned can always be monitored in practice, and some are more likely to be obtained as supplementary information. The result is a matrix of attack steps to be monitored and potentially suitable monitoring measures (cf. Table B.3). A selection was made from among the possible countermeasures that, as required, covers the attack steps mentioned. Instead of selecting a minimal covering set of sensors, a more sensible coverage of the steps was chosen, where some steps involve multiple measures of varying effort and perspective. For example, for the first step of compromising the cockpit control module (CCM) via WiFi or Bluetooth, three sensors were selected to monitor different layers of network communication (network stack for packet and encryption errors, network management for connection and address assignment errors, and service for application errors).

It is now interesting to examine the specific attack example from 2025. The vulnerability was uncovered by white hats and was quickly patched, which is why the actual damage was minimal. However, it could theoretically have led to the damage scenarios previously identified in TARA. The attack path in question is

B Exemplary application of the procedure to specify monitoring requirements

Table B.3: Selection of monitoring measures for keyless access item. ✓: chosen method in-vehicle, ★: chosen method in backend, •: suitable to monitor, ○: theoretically suitable based on mapping in Table A.6, but practically infeasible.

Step	Description	Observables	NIDS	Firewall	Flow collector	SIDS	Telemetry	OS	HIDS	Access control	Network Stack	Certificate Mgmt.	Service	Update Mgmt.	Execution Mgmt.	Network Mgmt.	CIM
1	Attacker compromises CCM via WiFi or Bluetooth to achieve remote code execution	1, 21, 6	•	•	•			•	○	✓		✓				✓	
4	Attacker sends malicious control signals from diagnostic port	1, 3, 5, 21	✓	•	•			•	✓	○				○	•	•	
6	Attacker compromises DCM via cellular network to achieve remote code execution	1, 21, 6	•	•	•			✓	•	✓		✓				•	
8	Backend is compromised allowing unauthorized command issuance	1, 4, 20, 21	○	○	○			•	★	★	○		★	○		○	
9	DCM receives malicious commands from authenticated connection and forwards them	1, 4, 20, 14	•	•	✓			•	•	✓	○			•	•	•	
10	Attacker gains access to user credentials or tokens enabling command issuance	3, 11, 14, 1, 4, 20	★	•	•			•	•	★	○			○	○	○	
11	Attacker intercepts or manipulates communication between vehicle and backend	1, 10, 21, 6, 9	•	•	•				○	○	✓	•	•				✓
12	CGW forwards unauthorized or unauthenticated commands	1	•	✓	✓			•	○	○						•	
D1	Attacker can start engine and drive away (theft of car)	7, 21				✓	✓				○		•			○	
D2	Attacker can open car (theft of personal belongings)	7, 21				•	✓				○		•			○	
D3	Attacker can access user account data	1, 11, 13	★	○	○			•	★	•	○						○

as follows [Cur25]: An attacker can gain online access to any vehicle through a poorly secured backend portal. To do this, a web search is used to find a valid Subaru administrator email address, to which, however, no access is required. The account is then taken over by resetting the password to one of the attacker's choosing. Furthermore, the two-factor authentication required when logging in with the new password can be bypassed. The admin account has extensive privileges, as it can assign new users to any vehicle. As a result, the hacker's account is assigned to another vehicle, allowing them to send commands to open and start the vehicle and retrieve its movement history.

When comparing the administrator account takeover to the attack steps modeled earlier, it aligns well with a combination of steps 8, "Backend is compromised allowing unauthorized command issuance," and 10, "Attacker gains access to user credentials or tokens enabling command issuance". If the mechanisms specified earlier are implemented optimally, each attack step presents an opportunity for detection. However, aggregation at the fleet level is key. Monitoring the backend is the earliest possible detection point at which an attack can potentially be intercepted during the reconnaissance or account creation phase, before a vehicle is actually accessed. Access control in the backend would flag logins from unexpected sources or at unusual times. The creation of multiple new accounts for many vehicles within a short time window would be an anomaly that could not be detected at the individual vehicle level but becomes apparent at the fleet level. NIDS, HIDS, and service monitoring in the backend could also indicate unusual access patterns and abuse of the admin API. Repeated operations on many vehicles could be particularly relevant, as these also become apparent at the fleet level.

In the vehicle, access control and flow monitoring in the Data Communication Module (DCM, Subaru's telematics unit) would show successfully authenticated sessions using accounts that were newly created or used for this vehicle. While this might still be normal for a single vehicle, across the fleet it quickly becomes an anomaly indicating misuse. Ultimately, the flow collector at the central gateway (CGW) can monitor the actual command to open the doors and start the engine sent from the DCM. Thanks to the monitoring capabilities in the backend and

DCM, it is likely that an attack can be detected early on, before any actual consequences occur. Even without backend detection, there is a chance to detect attackers before they cause harm. In the event of a successful attack and theft, the context information can be used to retroactively determine the causes of the attack, as local and timely correlations of successful thefts are likely. Overall, this example demonstrates effective monitoring of the vehicle fleet, which can lead to faster detection.

C Further background

C.1 Networking technologies

Controller Area Network

Due to the high number of electronic components in vehicles that need to exchange data, Bosch developed in the 1980s the CAN bus system that fulfills a reliable data exchange between ECUs [ISO15b]. CAN is a serial multi-master bus that supports different data rates with a maximum of 1 MBit/s (High-speed CAN [ISO16c]) and with CAN-FD¹ [HB12] up to about 5 - 8 MBit/s, depending on the physical transceiver. Furthermore, the extension CAN-XL was developed which allows for up to 10 MBit/s [HB20]. In-vehicle networks use multiple CAN buses, that are interconnected via gateways and a backbone network, if necessary [Nav+15]. Each frame contains a unique ID that defines the priority of a message.

On a specific CAN bus, a message with a specific ID is sent by no more than one ECU. All ECUs in the vehicle network can subscribe to any message ID. However, a gateway may be needed to route the messages, if sending and receiving ECU are attached to different buses. CAN messages are sent cyclically, which is mostly used to realize control and feedback loops with multiple ECUs or event-based, usually for interaction with the user (e.g. activation of the turn indicator, window regulator). Finally, the transmission of a specific ID can also be requested by another bus participant.

¹ CAN with Flexible Data Rate

Local Interconnect Network

The serial single-master bus system LIN has been designed as a sub-bus of CAN to connect simple sensors and actuators, for example a window lifter. Minimal costs for cabling and bus controllers are the main demands on this bus system, correspondingly low is the bandwidth, which is 20 kBit/s [ISO16d].

Automotive Ethernet

In addition to diagnostics and flashing of ECUs, Ethernet is increasingly used for communication within the vehicle, especially for high-bandwidth sensor data such as cameras. To meet the automotive real-time and embedded requirements such as freedom from interference, several networking standards for the physical layer were developed. With the 100BASE-T1 standard [80216a] data rates of 100 MBit/s are possible, with 1000BASE-T1 even 1 GBit/s [80216b]. Further increases in data rates above 2.5 GBit/s were standardized [IEEE20b]. To bridge the gap of data rates between CAN-FD and the 100BASE-T1 standard but still maintaining the low costs of CAN, the 10BASE-T1S physical layer was specified [IEEE20a; Zwe19]. In this gap, 10BASE-T1S is competing against CAN-XL.

C.2 Software architecture

Software in vehicles

Even in 2006, a vehicles ECUs realized more than 2000 individual functions with 10 million lines of code [Bro06]. However, the number of lines of software code in a car exploded from about 10 to 20 million in 2010 (e.g. former Mercedes-Benz S-class [Cha09]) to about 150 million in 2016 [Bur+18]. In 2020, about three million functions were implemented in a vehicle [Ant20]. Other estimates assume there will be up to 300 million lines of code in a car in 2030 [Dei+19]. Furthermore, the market of automotive software and electronics is expected to

grow at an annual rate of about seven percent between 2020 and 2030 up to 469 billion US dollars, whereas the automotive market in general will only grow about three percent per year in the same time [BDS19].

In the past, software was rigidly baked together with ECUs, so that a multitude of monolithic blocks, each with self-contained functionalities, consolidated the E/E architecture. As long as these functions essentially worked only with sensors and actors of their respective function domains, this worked well. In this approach, a supplier could deliver a black box that could be connected with other ECUs based on signals (cf. Section *Signal-based architecture* on page 308). For automated driving functions in particular, this domain separation no longer works - sensors and actors throughout the vehicle are involved. At the same time, conventional microcontrollers in ECU are no longer powerful enough to handle this volume of software and network communication. High-performance multi-core controllers have been introduced that can perform multiple, cross-domain functions as a generic computing platform.

With separation of hardware and software, independent development cycles and thus decreased time to introduce new software-based innovations to the market are expected [Nav+15; Bur+23]. The average annual growth rate in the development of operating systems and middleware, for example, is forecast to match the growth rate in development expenditure for automated driving functions [BDS19].

Software communication paradigms

Two different paradigms define the communication between the components and functions in the distributed vehicular software architecture.

- The first are signal-based architectures, where all communication relationships are defined during system design time. Data transmission is only controlled by the sender application by transmitting data as soon as it considers it necessary. It is a closely-coupled and static architecture.

- In contrast, service-oriented communication is loosely-coupled and more dynamic, because the communication relationships can be established at runtime by the middleware. The receiver, becoming a consumer, may control the data flow by requesting data of a provider.

a) Signal-based architecture In the signal-based paradigm each signal is a specific piece of information such as the engine revolution or the torque and is identified uniquely (e.g. by a name or a number). Multiple signals are grouped into a frame, which is identified by a frame identifier, whereby the order of the signals in a message with a specific identifier is fixed in the course of vehicle development. Figure C.1 shows an exemplary CAN frame consisting of three signals (torque, speed, signal 3) with different lengths.

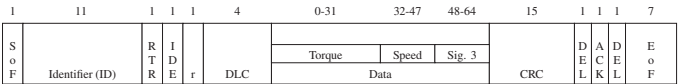


Figure C.1: The layout of a CAN frame, numbers indicate the field length in *bit*

Irrespective of whether data is currently required by a receiver in the network, the frame is sent. The network recipients have to decide if received data is relevant for themselves. On the physical layer, these signal-oriented frames are mostly transmitted on CAN, LIN or FlexRay since these follow broadcast-based transmission. In addition, the frames can be sent individually or grouped into multiple frames as a single packet via Ethernet (e.g. via UDP or TCP). This is especially relevant for gateway applications, e.g. when data must be sent from an ECU on CAN to an ECU in the Ethernet network. The rigid coupling of the communication also manifests itself in the fact that the software must explicitly take care of the correct identification of the data, and define the physical transmission path (e.g. CAN). All the fixed routes are documented in a Communication Matrix and integrated into the firmware of the ECUs. To ensure proper communication all ECUs need to be derived from the same version of the communication matrix.

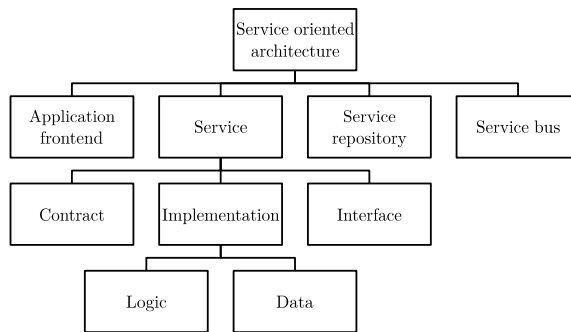


Figure C.2: Elements of a SOA according to Krafzig et al. [KBS05]

b) Service-oriented architecture In the automotive context, with the transition to Ethernet-based networks, a static pre-definition of information groups and sequences is not useful anymore, given that an Ethernet frame may contain up to 1500 Byte [Web15]. In contrast, SOA provides dynamic runtime communication links without static dependencies [Tis18].

In a service-oriented software architecture, the communication can follow a publish-subscribe pattern. Here, ECUs offer services as servers and may also subscribe as clients to the services of other ECUs. If new data gets available to a server, the subscribers get notified. In addition to the publish-subscribe pattern, the request-response and the fire-forget pattern are used. In both cases, the client calls a function of the server, however fire-forget does not return an answer to the client, or the client doesn't care about an answer.

In a SOA, each software component is encapsulated by a specified interface. The middleware takes care of pairing provider and consumer at runtime and converting data for the chosen transmission medium. While this approach requires additional software, networking overhead and computing capacity for the middleware, it allows for a more flexible distribution of software to hardware. In essence, the manual development effort of the static communication specification is countered with more computational and networking resources.

C.3 Security measures in vehicles

Based on the current cybersecurity culture of *Defense in depth*, cybersecurity measures that have been investigated in research and industry are illustrated based on a layered approach (cf. Figure 1.4).

Layer 1 - External interfaces:

In layer one, the vehicles' interfaces to the outside world are secured. This includes reducing the number of interfaces without neglecting customer needs [MV14]. The access for applications that are allowed to communicate with the "outside world" to the in-vehicle network should be restricted. Diagnostic access (with UDS) is secured with the two access control options *SecurityAccess* (UDS service 0x27) or *Authentication* (UDS service 0x29). Control of diagnostic access is important, especially for operations, that could change the desired behavior of the vehicle, e.g. flashing new firmware or manipulation of software in memory. The legacy *SecurityAccess* applies a challenge-response scheme to authorize the diagnostic device², while modern *Authorization* exchanges PKI³ certificates. Firewalls (cf. Section *Layer 3 - Network*: on page 311) can be used to filter network traffic through external interfaces.

Layer 2 - Applications:

The respective Methods to secure the operation of applications include access control mechanisms to grant or deny actions on resources (e.g. execute other applications, read or write files, access network). AUTOSAR, for example, specifies the IAM [AUT18]. Researchers analyzed different types of access control mechanisms, e.g. based on (dynamic) attributes or access control lists [Hug+19;

² called "tester"

³ Public Key Infrastructure (cf. glossary, page 328)

Ham20; Rum+19b]. Another option is the isolation of applications in different execution environments with separated OSs (virtualization, e.g. using PikeOS [Ber19]) or with containerization [RT19]. Operating systems can support security via ASLR⁴, which limits the impact of buffer overflow attacks [RT19]. Furthermore, applications can be secured with HIDS that observe and monitor applications and the OS of an ECU and detect deviations from normal behavior such as execution of uncommon system functions. This concept was analyzed in an AUTOSAR context [Tep+20], and a literature review [SEH21].

Layer 3 - Network:

The vehicles network can be secured with the methods on layer three. Communication with different safety criticality and disparate customer functions (e.g. body and comfort vs. powertrain) can be separated into different physical or virtual networks (VLAN, e.g. for zonal topology [Jun22]). In addition, firewalls can be used, e.g. as specified in AUTOSAR [AUT23a], to proactively block communication based on a set of predefined rules. If deployed on telematic control units, within the OBD connection, or other ECU with external connectivity, firewall rules to block external communication can be specified [Web18]. Within the vehicle, firewalls are deployed on hosts that connect different buses or Ethernet connections, such as gateways, zone controllers, or switches. In research, for example, the interaction of software and hardware for the implementation of firewalls was investigated [PSZ17]. Seifert and Obermaisser developed a secure gateway that detects and blocks attacks via the specified communication behavior [SO14]. An integrated switch firewall solution is offered in the industry by ETAS, for example⁵.

Next, (anomaly-based) NIDS identify deviations of the communication from the normal behavior of a network [GWS18] and are thus important for detecting

⁴ Address Space Layout Randomization

⁵ ETAS / ESCRYP T CysurGATE <https://www.etas.com/de/portfolio/cysurgate.php>

security incidents that exploit zero-day vulnerabilities. In contrast, rule-based NIDS detect known attacks or known uncommon behavior based on specified patterns. Due to the higher periodicity of in-vehicle traffic, NIDS are expected to show better detection results than is the case in enterprises [MV14].

Researched NIDS are mostly anomaly-based and several surveys discuss NIDS for CAN [Al+19]. The approaches detect attacks on CAN based on message characteristics (e.g., formality, location, plausibility of messages) [MGF10], communication properties such as cycle time [SMC17], correlated signals (e.g., speed, location, signal courses) [Was+17], characteristics of the physical channel (e.g., voltage courses of transmitted bits) [Rum+19a]. This work distinguishes between methods that monitor the entire network traffic based on characteristics (NIDS) and methods that monitor the trend or correctness of specific individual information (SIDS). Payload-based approaches are usually SIDS, because they learn the behavior of signals in the vehicle, e.g. [Web19]. Recently, Lampe and Meng [LM23] discuss IDS for automotive in a broader context, also surveying Ethernet- and SOA-focused approaches. IDS for CAN may be extended to an IPS by sending error frames [OHB19] or blocking messages. NIDS for CAN can be also realized within a CAN transceiver, enabling the detection of flooding, or malicious CAN identifiers⁶.

Besides detection mechanisms, a *zero trust* approach on network defense layer can be realized with PNAC⁷ on Ethernet networks, if the switches support the respective protocol IEEE 802.1X [Jun22]. With PNAC, no hardware device in the network is trusted (has access) before verification of the devices certificate. After verification, the communication is encrypted with MACsec⁸ (cf. Layer 4).

⁶ NXP TJA115x Secure CAN Transceiver <https://www.nxp.com/docs/en/fact-sheet/SECURCANTRLFUS.pdf>

⁷ port-based network access control (cf. glossary, page 328)

⁸ Media Access Control Security (cf. glossary, page 327)

Layer 4 - Data:

Encryption and authentication mechanisms are the major elements of layer four for securing the data. Here, the AUTOSAR-specified SecOC [AUT14] mechanism to authenticate CAN and Ethernet communication and the internet-standards for encryption and authentication of Ethernet communication (MACsec, IPSec, TLS, DTLS⁹) are listed (cf. Table C.1). Proposals from research for CAN authentication were evaluated by Nowdehi et al. [NLO17]. Note, that if an attacker has compromised a host, access to key material is usually possible, enabling the attacker to send valid messages [MV14]. Data that is not in motion (i.e. configuration data and databases in the vehicle and backend) can be secured with encryption mechanisms and access control (authentication and authorization), or may be stored in secure hardware (cf. Layer 5). External interfaces should incorporate secured transmission as well (cf. Layer 1) [Web18]. This especially includes encrypted communication channels to the manufacturers' servers, for example with VPN technology [Sto21].

Layer 5 - ECU:

The innermost layer consists of hardware security measures and procedures to manage keys and signatures stored in each ECU that are required for authentication or encryption of communication and data. The roots of trust are tamper-proof modules of microcontrollers that provide the hardware for e.g. encryption and storing of keys to the applications on that ECU [Web18; RT19]. Based on this hardware anchor firmware code such as the bootloader and the update mechanism is authenticated and maintains a chain of trust with secure boot and secure update. [Web18; Bar19]

⁹ Datagram Transport Layer Security (cf. glossary, page 325)

Table C.1: Common automotive and security protocols within the ISO/OSI model. (Diagnostics specific protocols are italicized, security protocols are bold)

ISO/ OSI Layer	Automotive Protocols									
7	<i>UDS, OBD</i>	SecOC	SecOC	SOME/IP	<i>UDS, OBD, DoIP</i>	DDS, DDS Security	DHCP	ICMP	AVB /TSN	ARP
6										
5			TLS/DTLS							
4	<i>ISO TP, SAE J1939</i>	TCP/UDP						ICMP	AVB /TSN	ARP
3		IPv4, IPv6, IPSec								
2	CAN	Ethernet MAC, VLAN, MACSec								
1		Automotive Ethernet PHY (100BASE-TX, 100BASE-T1, 1000BASE-T1, Multi-Gig)								

Acronyms

AAP AUTOSAR Adaptive platform. 20, 21, 33, 161, 178, 179, 196, 325, 326, *Glossary*: AUTOSAR Adaptive platform

ACF Applied Collection Framework. 73, 104, 115, 149, 150, 334

ACP AUTOSAR Classic platform. 20, 21, 33, 161, 170, 172, 325, *Glossary*: AUTOSAR Classic platform

ADAS Advanced Driver Assistance Systems. 117, 329, *Glossary*: Advanced Driver Assistance Systems

ADORE Automated Driving Open Research. 225, 239, *Glossary*: Automated Driving Open Research

ADS Automated Driving System. 38, 39, 183

AI Artificial Intelligence. 91, 203–206, 270, 327

API Application Programming Interface. 20, 93, 169, 192, 260, *Glossary*: Application Programming Interface

ARP Address Resolution Protocol. 17

ASIL Automotive Safety Integrity Level. 36, 37, *Glossary*: Automotive Safety Integrity Level

ASLR Address Space Layout Randomization. 311

ASRG Automotive Security Research Group. 82, 83

ATM Automotive Threat Matrix. 83, 147

ATT&CK® Adversarial Tactics, Techniques, and Common Knowledge. 74, 81–83, 92, 104, 105, 107, 116, 117, 160, 177, 193, 281, 283–291, 338

Auto-ISAC Automotive Information Sharing and Analysis Center. 66, 81–83, 175

AUTOSAR AUTomotive Open System ARchitecture. 20, 30, 33, 79, 89, 93, 106, 114, 169, 171, 172, 176–178, 263, 324–326, 329, *Glossary*: AUTomotive Open System ARchitecture

AVB Audio/Video Bridging. 16

CA Certification Authority. *Glossary*: Certification Authority

CAN Controller Area Network. 1, 5, 11, 15, 18, 21, 67, 78, 79, 90, 106, 161, 172, 200, 305, 306, 308, 312, 313, 328, 336

CAN-FD CAN with Flexible Data Rate. 305, 306

- CAPEC** Common Attack Pattern Enumeration and Classification™. 81, 84, 92, *Glossary*: Common Attack Pattern Enumeration and Classification™
- CI/CD** Continuous integration / Continuous deployment. 25, 170, 173, 217, 247, *Glossary*: Continuous integration / Continuous deployment
- CIM** Cybersecurity Information Management. 88, 106, 177, 184, 187, 235–238, 240, 241, 248, 257, 264, 335
- CMS** cybersecurity monitoring sensor. 73, 74, 76, 78, 88, 89, 104–106, 115, 123–125, 138, 146, 158, 161, 234, 236, 262, 337
- CNCF** Cloud Native Computing Foundation. 24, 241
- COD** Current Operational Domain. 38, 39, 79, 169, 182–184, 200, 264
- CPE** Common Platform Enumeration. 92, *Glossary*: Common Platform Enumeration
- CPU** Central Processing Unit. 21–24
- CSA** Cybersecurity Situation Awareness. 63, 75, 106, 107, 115, 117, 124, 127, 167, 192, 193, 195, 200
- CSAF** Common Security Advisory Framework. 81
- CSI** cybersecurity information. 73, 88, 91, 115, 133, 136, 137, 145, 198, 201
- CSMS** Cyber Security Management System. 6, 7, 65, 103
- CTI** Cyber Threat Intelligence. 75, 80–82, 85, 92, 106, 107, 112, 117, 141, 147, 166, 174, 188, 193, 202–204, 268, 270
- CVE** Common Vulnerabilities and Exposures. 9, 70, 86, 91, 107, 126, 173, 202, 247, 252, 324, *Glossary*: Common Vulnerabilities and Exposures
- CVE-ID** Common Vulnerabilities and Exposures identifier. *Glossary*: Common Vulnerabilities and Exposures identifier
- CVSS** Common Vulnerability Scoring System. 45, 70, 92, 137, 247, *Glossary*: Common Vulnerability Scoring System
- CWE** Common Weakness Enumeration. 70, 91, *Glossary*: Common Weakness Enumeration
- DDS** Data Distribution Service. 21, 252, *Glossary*: Data Distribution Service
- DevOps** Development and Operations. 25, 68, 121, *Glossary*: Development and Operations
- DevSecOps** Development, Security, and Operations. 25

- DFD** Data flow diagram. 44, 47, 144, 151, 153, 218, 297
- DHCP** Dynamic Host Configuration Protocol. 17, 92, 105, 176, 240, *Glossary*: Dynamic Host Configuration Protocol
- DLT** Diagnostic Log and Trace. 79, 105, 178, 179, *Glossary*: Diagnostic Log and Trace
- DNS** Domain Name System. 17, 92, 105, 176, 240, *Glossary*: Domain Name System
- DTC** Diagnostic Trouble Code. 169, 177, 200, 327
- DTLS** Datagram Transport Layer Security. 313, 330, *Glossary*: Datagram Transport Layer Security
- E/E architecture** electrical and electronic architecture. 2, 13, 20, 24, 29, 33, 53, 55, 62, 116, 169, 221
- ECU** Electronic Control Unit. 1, 10, 17, 20, 21, 24, 26, 33, 69, 78, 105, 199, 210, 215, 293, 305–309, 311, 329
- FMEA** Failure Mode and Effects Analysis. 35, 36
- FOTA** Firmware-over-the-air. 33
- GNN** Graph Neural Network. 205
- GNSS** Global Navigation Satellite System. 73, *Glossary*: Global Navigation Satellite System
- GQL** Graph Query Language. 111, 190
- HARA** Hazard Analysis and Risk Assessment. 35–38, 270
- HAZOP** Hazard and Operability Study. 36
- HIDS** Host-based Intrusion Detection System. 74, 78, 176, 177, 234–236, 254, 257, 292, 311, *Glossary*: Host-based Intrusion Detection System
- HSVL** High Speed Video Link. 15, *Glossary*: High Speed Video Link
- HTTPS** Hypertext Transfer Protocol Secure. 251
- IAM** Identity and Access Management. 79, 310, *Glossary*: Identity and Access Management
- ICM** InnovationCampus Future Mobility. 222, *Glossary*: InnovationCampus Future Mobility
- ID** identifier. 172, 305
- IDPS** Intrusion Detection and Prevention System. 79, 128, *Glossary*: Intrusion Detection and Prevention System
- IDS** Intrusion Detection System. 11, 71–73, 76, 78, 79, 89, 106, 161,

- 169, 199, 201, 205, 312, 326, 327, 329, *Glossary*: Intrusion Detection System
- IdsM** Intrusion detection system Manager. 89, 93
- IdsR** Intrusion detection system Reporter. 89
- IETF** Internet Engineering Task Force. 89
- IP** Internet Protocol. 16, 17, 92, 172, 200, 325, 329
- IPS** Intrusion Prevention System. 11, 78, 312, 326, *Glossary*: Intrusion Prevention System
- IPSec** Internet Protocol Security. 11, 313, 330
- ISAC** Information Sharing and Analysis Center. 51, 81, 107, 142
- ISO** International Organization for Standardization. 26, 34, 39
- IT** Information Technology. 1, 5, 10, 17, 24, 61, 62, 64, 67, 70, 91, 94, 147, 325, *Glossary*: Information Technology
- I²C** Inter-Integrated Circuit. 220
- J-Auto-ISAC** Japan Automotive Information Sharing and Analysis Center. 81
- JSON** JavaScript Object Notation. 81, 89, 90, 106, 244, 247, 281
- KG** Knowledge Graph. 91, 92, 127, 135, 165–167, 170–172, 174, 175, 182, 188, 190–192, 194, 199, 203–206, 217, 219, 220, 236, 241, 242, 244–249, 254, 262, 263, 268, 270, 271, 281, 335, *Glossary*: Knowledge Graph
- LIN** Local Interconnect Network. 15, 306, 308
- LLM** Large Language Model. 204–206, *Glossary*: Large Language Model
- LPG** Labeled Property Graph. 109, 167, 188, 189, 194, *Glossary*: Labeled Property Graph
- MAC** Media Access Control. 16, 172, 200
- MACsec** Media Access Control Security. 312, 313, *Glossary*: Media Access Control Security
- MBSE** model-based systems engineering. 2, 27, 31, 115, 262, 263
- MFA** Multi-factor authentication. 264, *Glossary*: Multi-factor authentication
- MISP** Malware Information Sharing Platform. 81
- MRM** minimal risk maneuver. 39, 56
- NHTSA** National Highway Traffic Safety Administration. 82
- NIDS** Network-based Intrusion Detection System. 78, 94, 150,

- 176, 292, 311, 312, *Glossary*: Network-based Intrusion Detection System
- NIST** National Institute of Standards and Technology. 70–72, 324
- Nmap** Network Mapper. *Glossary*: Network Mapper
- NSM** network security monitoring. 76, 78, 106, 112, 115, 124–126
- OBD** On-Board Diagnostics. 5, 32, 170, 197, 311, *Glossary*: On-Board Diagnostics
- OCI** Open Container Initiative. 23, *Glossary*: Open Container Initiative
- ODC** Operational Design Conditions. 38, 39, 105, 125, 169, 181, 182, 200
- ODD** Operational Design Domain. 38, 39, 79, 91, 117, 181–183, 200, 264, 270
- OEM** Original Equipment Manufacturer, synonym for vehicle manufacturer. 1, 5, 12, 26, 32, 53, 55, 62, 68, 69, 108, 119, 122, 147, 149, 167, 171, 174, 175, 198, 203, 210, 213, 261, 334
- OMG** Object Management Group. 325
- OODA** observe, orient, decide, act. 70
- OS** Operating System. 1, 20, 21, 23, 34, 76, 78, 112, 161, 170, 176, 177, 185, 199, 203, 228, 233, 292, 293, 311, 324
- OSI** Open Systems Interconnection. 18, 20, 76, 78, 329, 333, 337
- OTA** over-the-air. 3, 11, 13, 32–34, 55, 56, 127, 170, 187, 250
- OWASP** Open Web Application Security Project. 82, 325
- PID** Proportional–integral–derivative. 225
- PKI** Public Key Infrastructure. 310, 329, *Glossary*: Public Key Infrastructure
- PNAC** port-based network access control. 312, 327, *Glossary*: port-based network access control
- PoE** Power over Ethernet. 220, *Glossary*: Power over Ethernet
- POSIX™** Portable Operating System Interface. 21, 161, 170, 178, 324, *Glossary*: Portable Operating System Interface
- PTP** Precision Time Protocol. 16
- RAM** Random Access Memory. 21, 22, *Glossary*: Random Access Memory
- RDF** Resource Description Framework. 109, *Glossary*: Resource Description Framework
- RFLP** Requirements - Functional - Logical - Physical. 28, 151,

- Glossary: Requirements - Functional - Logical - Physical*
- RL** Reinforcement Learning. 204, 206
- ROS 2** Robot Operating System 2. 21, 172, 178, 180, 217, 221, 235, 236, 238, 240, 252, 325, *Glossary: Robot Operating System 2*
- SAE** Society of Automotive Engineers. 37, 39
- SBOM** Software Bill of Materials. 66, 67, 85, 173, 247, 325
- SCA** software composition analysis. 85
- SDN** Software-defined Networking. 94
- SDV** software-defined vehicle. 1, 15, 20, 62, 68, 105, 112, 127, 167–169, 176–179, 185, 191, 199, 208, 210
- SecOC** Secure Onboard Communication. 11, 313, *Glossary: Secure Onboard Communication*
- SerDes** Serializer/De-Serializer. 15, 325, *Glossary: Serializer/De-Serializer*
- SEv** Security Event. 79, 89, 105, 106
- SIDS** Signal-based Intrusion Detection System. 176, 234, 235, 292, 312, *Glossary: Signal-based Intrusion Detection System*
- SIEM** Security Information and Event Management. 63, 65, 67, 72, 74, 88, 90, 92, 106, 112, 142, 201, 205
- SOA** Service-oriented Architecture. 2, 16, 21, 22, 53, 78, 125, 171, 176–180, 184, 194, 216, 217, 309, 312, 325, 329, 336, *Glossary: Service-oriented Architecture*
- SOC** Security Operations Center. 63–65, 67, 70, 75, 80, 88, 92, 147, 265
- SOME/IP** Service-oriented Middleware over IP. 21, *Glossary: Service-oriented Middleware over IP*
- SOP** Start of production. 40
- SOTIF** Safety Of The Intended Functionality. 34, 37, 38, 67, 80, 101, 270, 271
- SPLE** Software Product Line Engineering. 30, 171, 196, 262
- SQL** Structured Query Language. 108, 111, 190
- STIX** Structured Threat Information eXpression. 81, 244, 281
- STRIDE** Spoofing - Tampering - Repudiation - Information disclosure - Denial of service - Elevation of privilege. 44, 47, 83, 151–153, 226
- SUMS** software update management system. 33
- SysML** Systems Modeling Language. 27

- TARA** Threat Analysis and Risk Assessment. 10, 35, 37, 40, 42, 43, 45, 46, 50, 54, 68, 69, 86, 100–105, 107, 115, 121–123, 131, 141, 143–152, 154, 158, 171, 172, 180, 182, 194, 197, 202, 218, 249, 250, 261, 263, 267, 270, 273, 297, 298, 300, 333, 338
- TCP** Transmission Control Protocol. 17, 76, 187, 238, 240, 308, 329
- TCU** Telematic Control Unit. 218, 228, 234, 235, 249
- TLS** Transport Layer Security. 11, 106, 114, 126, 129, 187, 214, 313, 325, 330, *Glossary*: Transport Layer Security
- TSN** Time Sensitive Networking. 16
- TTP** tactics, techniques, and procedures. 80, 82, 83, 104, 116, 117, 119, 121, 123, 125, 146, 147, 151, 152, 155, 156, 193, 203, 281, *Glossary*: tactics, techniques, and procedures
- UDP** User Datagram Protocol. 17, 76, 240, 308, 325, 329
- UDS** Unified Diagnostic Services. 11, 33, 170, 210, 310, *Glossary*: Unified Diagnostic Services
- UML** Unified Modeling Language. 27, 193, 194
- UN R155** UN Regulation No. 155. 6–8, 10, 12, 53, 54, 58, 61, 65, 69, 83, 84, 97, 102, 114, 261
- USB** Universal Serial Bus. 15
- V2X** Vehicle to Everything. 37, 222
- VIN** Vehicle Identification Number. 236, 248, 292
- VLAN** Virtual Local Area Network. 11, 16, 311, *Glossary*: Virtual Local Area Network
- VM** virtual machine. 22, 23, 105, 177, 220, 221, 254, 255
- VPN** Virtual private network. 126, 129, 167, 187, 188, 214, 240, 248, 313, *Glossary*: Virtual private network
- VSOC** Vehicle Security Operations Center. 64, 66, 67, 75, 80, 88–90, 94, 97, 103, 106–108, 110, 115, 117, 119, 121–123, 125, 127–129, 131, 142, 146–148, 165, 167, 174, 175, 181, 184–192, 208, 210, 211, 213–215, 219, 236, 240, 248, 261, 264–266, 268, 269, 334
- XML** Extensible Markup Language. 171, 172, 244, 263
- YAML** Yet Another Markup Language. 244, 247, 263, *Glossary*: Yet Another Markup Language

Glossary

Advanced Driver Assistance Systems ADAS (Advanced Driver Assistance Systems) are software and hardware systems to realize driving automation of SAE levels one or two, such as lane keep assistance, adaptive cruise control, and adaptive headlight systems.

Application Programming Interface An API (Application Programming Interface) is a defined interface of a software to interact with or be controlled by another software.

AutoDevSafeOps Research project "Integrierte Entwicklung und Betrieb von sicheren Automotive-Systemen (MANNHEIM-AutoDevSafeOps), funded by the German Federal Ministry of Education and Research. The project is researching a holistic DevOps approach with integrated safety engineering with the aim of enabling modular updates including the associated safety processes and methods. https://www.softwaresysteme.dlr-pt.de/media/content/01IS22087_Projektblatt_MANNHEIM-AutoDevSafeOps.pdf.

Automated Driving Open Research Eclipse ADORe (Automated Driving Open Research) "is a modular software library and toolkit for decision making, planning, control and simulation of automated vehicles " [FT] developed by the German Aerospace Center (DLR).

AUTomotive Open System ARchitecture The AUTOSAR (AUTomotive Open System ARchitecture) foundation is an organization that defines open standards and a development methodology for automotive software architectures.

Automotive Safety Integrity Level "One of four levels to specify the item's or element's necessary ISO 26262 requirements and safety measures to apply for avoiding an unreasonable risk (3.176), with D representing the most stringent and A the least stringent level" [ISO18b]. The ASIL is determined for hazardous events based on the classification of severity

of potential harm, probability of exposure to the respective operational situation, and controllability [ISO18c].

AUTOSAR Adaptive platform The AUTOSAR Adaptive platform is a service-oriented software architecture standard based on POSIX™ OS, targeting microprocessors and compute-intensive tasks.

AUTOSAR Classic platform The AUTOSAR Classic platform is a signal-based software architecture standard, targeting real-time requirements and microcontrollers.

Certification Authority Organization that provides a (root) CA certificate as the root of trust in a PKI and handles requests for new certificates.

Common Attack Pattern Enumeration and Classification™ CAPEC (Common Attack Pattern Enumeration and Classification) is a list of common ways adversaries exploit vulnerabilities, structured in a hierarchy and maintained by MITRE (<https://capec.mitre.org/about/index.html>).

Common Platform Enumeration CPE (Common Platform Enumeration), <https://nvd.nist.gov/products/cpe>, is a standard for a uniform formal naming convention for information technology systems, platforms and software packages, maintained by NIST and usually identifies products affected of CVE entries.

Common Vulnerabilities and Exposures The CVE (Common Vulnerabilities and Exposures) is a public database and standardized naming system for vulnerabilities in software and hardware, maintained by MITRE, <https://www.cve.org/>.

Common Vulnerabilities and Exposures identifier Unique identifier for each entry in the CVE of the format CVE-<year>-<consecutive number>.

Common Vulnerability Scoring System CVSS (Common Vulnerability Scoring System), <https://www.first.org/cvss/specification-document> "provides a way to capture the principal characteristics of a vulnerability and produce a numerical score reflecting its severity" [For23]. ISO/SAE 21434 recommends using the CVSS base metrics, which score attack vector, attack complexity, attack requirements, required privileges, and user interaction.

Common Weakness Enumeration CWE (Common Weakness Enumeration), <https://cwe.mitre.org/>, is a database and categorization maintained by MITRE with common weaknesses in hardware and software, each with a unique identifier, that can lead to vulnerabilities.

Continuous integration / Continuous deployment Tool and method to automate compiling, testing and rolling out when software components change.

CycloneDX CycloneDX is a BOM (Bill of Materials, cf. SBOM) standard defined by OWASP .

Data Distribution Service DDS (Data Distribution Service) is an AUTOSAR and OMG¹⁰ standard to implement SOAs. DDS is used in different industries [Obj04], and is the default middleware in ROS 2.

Datagram Transport Layer Security DTLS is TLS over datagram transport, especially UDP..

Development and Operations DevOps (Development and Operations) is a cooperation model of software development and IT operation based on automation of the deployment after software changes and fast feedback.

Diagnostic Log and Trace DLT (Diagnostic Log and Trace) is an AUTOSAR-specified protocol [AUT16a] implemented by ACP and AAP to provide an interface to applications for logging, and tracing the AUTOSAR middleware.

Domain Name System DNS (Domain Name System) is the worldwide, hierarchical directory system to resolve names of servers to their IP addresses and vice versa.

Dynamic Host Configuration Protocol DHCP (Dynamic Host Configuration Protocol) is a UDP broadcast based protocol to assign static or dynamic IP addresses to hosts.

Global Navigation Satellite System Generic term for satellite-based navigation, for example the Global Positioning System (GPS) or Galileo.

High Speed Video Link Proprietary networking technology, similar goals as SerDes.

¹⁰ Object Management Group

Host-based Intrusion Detection System "IDSs which operate on information collected from within an individual computer system. This vantage point allows host-based IDSs to determine exactly which processes and user accounts are involved in a particular attack on the Operating System. Furthermore, unlike network-based IDSs, host-based IDSs can more readily "see" the intended outcome of an attempted attack, because they can directly access and monitor the data files and system processes usually targeted by attacks" [NIS19].

Identity and Access Management IAM (Identity and Access Management) is an AUTOSAR-specified [AUT18] access control mechanism that limits application's usage of AAP resources like network and data access based on digitally signed application manifest documents.

Information Technology Gartner defines: "IT (Information Technology) infrastructure is the system of hardware, software, facilities and service components that support the delivery of business systems and IT-enabled processes." Source: <https://www.gartner.com/en/information-technology/glossary/it-infrastructure>, accessed on 2024-12-03.

InnovationCampus Future Mobility The InnovationCampus Future Mobility (ICM) is a joint research platform of the Karlsruhe Institute of Technology (KIT) and the University of Stuttgart funded by the Baden-Württemberg Ministry of Science, Research and Arts since July 1, 2019 for cutting-edge research into sustainable mobility and production of tomorrow. <https://www.icm-bw.de/en/>.

Intrusion Detection and Prevention System Combination of IDS and IPS.

Intrusion Detection System "A security service that monitors and analyzes network or system events for the purpose of finding, and providing real-time or near real-time warning of, attempts to access system resources in an unauthorized manner" [NIS19].

Intrusion Prevention System "Software that has all the capabilities of an intrusion detection system and can also attempt to stop possible incidents" [NIS07].

Knowledge Graph A KG (Knowledge Graph) is a database that builds on a graph-structured data model.

Labeled Property Graph A LPG (Labeled Property Graph) is a directed graph database model where edges and vertices can have key-value-pairs of properties, edges have a label, and vertices have one or more labels.

Large Language Model Class of AI models trained on a large corpus of (textual) data, usually based on a transformer neural network architecture.

Media Access Control Security MACsec (Media Access Control Security) is an OSI layer two encryption mechanism for integrity and authenticity defined in IEEE 802.1AE that can be used together with PNAC.

middleware Software that abstracts the communication endpoints from the network topology and thus enables transparent communication between software components at network and host level.

MITRE The Mitre Corporation (or MITRE) is an US-based non-profit organization managing federally funded research and development centers, including cybersecurity.

Multi-factor authentication MFA (multi-factor authentication) is a method to authorize access to digital resources using at least two independent factors to authenticate a user, including knowledge (passwords), inherence (biometry), ownership (smartphone, smartcard, token).

Network Mapper Open source software for network discovery and security auditing via scanning.

Network-based Intrusion Detection System "IDSs which detect attacks by capturing and analyzing network packets. Listening on a network segment or switch, one network-based IDS can monitor the network traffic affecting multiple hosts that are connected to the network segment" [NIS19].

On-Board Diagnostics OBD (On-Board Diagnostics) is a standard for the diagnostic plug connector, the messaging format and DTCs.

Open Container Initiative Project of the Linux Foundation to standardize containerization, <https://opencontainers.org/>.

port forwarding Port forwarding refers to the redirection of a client connecting to a specific port (of a router or firewall) to another computer (server). The source addresses of incoming packets are replaced to forward the

request to the actual server. When the server responds, the destination address is replaced to forward it to the original client (cf. Wikipedia, https://en.wikipedia.org/wiki/Port_forwarding).

port-based network access control OSI layer two authentication and access control on Ethernet switches defined in IEEE 802.1X [Jun22].

Portable Operating System Interface POSIX™ (Portable Operating System Interface) is a "standard operating system interface and environment, including a command interpreter (or “shell”), and common utility programs to support applications portability at the source code level" defined in IEEE/Open Group standard 1003.1-2024 [IEE24].

post-production phase "the period in which a vehicle type is no longer produced until the end-of-life of all vehicles under the vehicle type" [EEC21a].

Power over Ethernet PoE (Power over Ethernet) is a combined data and power supply of devices via a standard Ethernet cable. PoE-capable devices do not need a separate power supply.

Public Key Infrastructure PKI (Public Key Infrastructure) issues, distributes and verifies digital certificates via asymmetric cryptography to sign and encrypt communication over a network.

Random Access Memory Memory of a computer that can be read and written in any order.

Requirements - Functional - Logical - Physical RFLP (Requirements - Functional - Logical - Physical) is a view-based development methodology based on multiple granularities and models for each of the four views.

Resource Description Framework RDF (Resource Description Framework) is a data model to describe directed graphs with formal semantics based on triple statements (subject-predicate-object), originally developed to formalize metadata for the internet.

Robot Operating System 2 A set of libraries and tools to develop (service-oriented) software for robots and automated vehicles.

Secure Onboard Communication SecOC (Secure Onboard Communication) [AUT14] is an AUTOSAR-specified mechanism to authenticate CAN and Ethernet communication.

Serializer/De-Serializer SerDes (Serializer/De-Serializer) is an automotive networking protocol and physical standard developed for asymmetric high-data rate point-to-point connections at low costs, such as connecting ADAS sensors to ECU, or ECU to displays.

service ISO/IEC 18384-1:2016 defines a service to be a "logical representation of a set of activities that has specified outcomes, is self-contained, may be composed of other services, and is a "black box" to consumers of the service" [ISO16a] .

Service-oriented Architecture The ISO/IEC 18384-1:2016 "Reference Architecture for Service Oriented Architecture (SOA RA)" defines SOA as: "architectural style that supports service orientation and is a paradigm for building business solutions." Moreover, service orientation is defined as an "approach to designing systems in terms of services and service-based development" [ISO16a].

Service-oriented Middleware over IP SOME/IP [AUT16b] (Service-oriented Middleware over IP) is an AUTOSAR standard to implement SOAs, specifically developed for the automotive industry. It supports transmission via TCP or UDP over an IP-based communication. SOME/IP is used in production vehicles, for example in Mercedes [Sch+20a] and Renault cars, summing up to several million vehicles on the road [VVS24].

Signal-based Intrusion Detection System IDSs which detect attacks by capturing and analyzing one or more communication signals.

tactics, techniques, and procedures see Definition 4.6 on page 83.

Transport Layer Security TLS (Transport Layer Security) is a cryptographic protocol, usually over a TCP connection, to provide confidentiality, integrity, and authenticity of communication using certificates and a session-specific shared key for encryption.

Unified Diagnostic Services UDS (Unified Diagnostic Services) is a manufacturer-independent automotive protocol on OSI layers 5 to 7 specified in ISO 14229 [ISO18d] defining standard diagnostic interfaces for all ECUs, including a secured service access via a challenge-response or PKI-based authentication.

Virtual Local Area Network A VLAN (Virtual Local Area Network) is a logically separated subnetwork of a physical Ethernet, that is created by VLAN-capable switches, which ensure not to forward frames to another VLAN, most commonly implemented with IEEE 802.1Q.

Virtual private network A VPN (Virtual private network) extends private networks over other, potentially untrusted, networks to hosts without direct access to the private network, using protocols for tunneling, including TLS, DTLS, IPSec and WireGuard.

Yet Another Markup Language YAML (YAML Ain't Markup Language, or originally Yet Another Markup Language) is a human-readable serialization language supporting scalars, lists, and maps, commonly used for configuration files.

Nomenclature

The following list describes several symbols that are used within the proposed development procedure (cf. Chapter 8).

δ, Δ	damage scenario(s)
γ, Γ	logical component(s)
$\mathcal{C}$	Catalog of tactics and techniques
$\mathcal{M}_{\mathcal{S}, \mathcal{O}}$	sensors-to-observables mapping
$\mathcal{M}_{\mathcal{T}, \mathcal{O}}$	techniques-to-observables mapping
ω, Ω	observation(s)
π, Π	attack path(s)
$\pi^{\mathcal{T}}$	technique-based attack path
θ, Θ	tactic(s)
v, Υ	threat scenario(s)
$\zeta : \mathcal{R} \mapsto \Gamma$	allocation of requirements
a, A	attack step(s)
A_G	attack steps subject to monitoring
D	Number of damage scenarios δ_d of an item

$G : A \mapsto \{0, 1\}$	Assignment of monitoring goals
I	Number of tactics θ_i of a catalog $\mathcal{C}$
J	Number of techniques t_j of a catalog $\mathcal{C}$
K	Number of threat scenarios v_k of an item
L	Number of observables o_l considered in the mapping $\mathcal{M}_{\mathcal{T}, \mathcal{O}}$
N	Number of attack steps a_n of an attack path π_p
$o, \mathcal{O}$	observable(s)
P	Number of attack paths π_p of an item
$r, \mathcal{R}$	monitoring requirement(s)
$s, \mathcal{S}$	sensor(s)
$t, \mathcal{T}$	technique(s)
$T : A \mapsto \mathcal{T}$	Classification of attack steps
$T_\delta : \Delta \mapsto \mathcal{T}$	Classification of damage scenarios
$T_v : \Upsilon \mapsto \mathcal{T}$	Classification of threat scenarios
U	Number of observations ω_u considered for the item
W	Number of sensors s_w considered in the mapping $\mathcal{M}_{\mathcal{S}, \mathcal{O}}$
$x_{s_w, \gamma}$	decision variable for optimization and allocation

List of Figures

- 1.1 The connected car and its interfaces serving as attack vectors. . . . 2
- 1.2 Distribution of attack vectors for incidents in automotive systems in 2023. 4
- 1.3 Number of new automotive security vulnerabilities 2013-2022 (CVE entries). 9
- 1.4 Cybersecurity measures in the vehicle in a defense in depth approach. 10
 - 2.1 Qualitative cost comparison of automotive networking technologies. 16
 - 2.2 Domain architecture topology "Base2017". 18
 - 2.3 Domain centralized architecture in the STAR3 architecture. 19
 - 2.4 Centralized in-vehicle server architecture in the E³ architecture. . . 19
 - 2.5 Schematic comparison of different software architectures in the OSI model. 20
 - 2.6 Comparison of virtualization approaches. 22
 - 2.7 Common phases in a DevOps lifecycle model. 25
 - 2.8 The system development sub-model of the V-model '97. 28
 - 2.9 Cybersecurity engineering according to ISO/SAE 21434 and alignment of the work products and clauses in the V-model. 41
 - 2.10 The concept and product development phases in detail. 42
 - 2.11 TARA outcome: artifacts and their multiplicity. 42
 - 2.12 The HEAVENS 2.0 TARA methodology. 47
 - 2.13 Nomenclature for the position of attack steps in attack trees. 47
 - 2.14 Processes covering continuous and post-development activities in ISO/SAE 21434. 49
 - 2.15 Development of sources used for cybersecurity monitoring 2023 and 2024. 51
- 3.1 Overview of this dissertation's idea. 57

4.1	Mapping the UN R155 tasks and research questions of this thesis to ideas and requirements.	61
4.2	Structure of the considered related work in cybersecurity operations and alignment with this thesis' ideas.	62
4.3	T-Systems VSOC approach.	66
4.4	Support organization and information flow for registering car security incidents at an OEM.	68
4.5	IT security incident management and handling processes.	72
4.6	The Applied Collection Framework.	74
4.7	Overview of sources for monitoring.	77
4.8	Common structure of SIEM systems.	89
4.9	Information contained in an IETF syslog message.	90
5.1	Qualitative comparison of ISO/SAE 21434, CMU/SEI-2004-TR01, NIST SP 800-61, and ISO/IEC 27035 as an overall cybersecurity operations process.	116
6.1	Overview of the proposed cybersecurity operations approach. . . .	120
6.2	Overview of the procedural concept for cybersecurity operations. .	121
6.3	Overview of the proposed procedure to specify monitoring requirements.	123
6.4	Overview of the monitoring concept.	124
6.5	Overview of the response concept.	128
7.1	Proposed cybersecurity operations process.	132
7.2	Interfaces to the TARA management process.	143
7.3	DevOps-inspired workflow for the continuous improvement process.	148
8.1	Mapping ISO/SAE 21434's development phase to the ACF.	150
8.2	Overview of the extended ISO/SAE 21434 concept phase to identify data feeds.	151
8.3	Modified TARA with threat information.	153
8.4	Visual overview of the entire procedure for specifying monitoring requirements.	159
9.1	Overview of the design of holistic monitoring.	166
9.2	Taxonomy of cybersecurity monitoring sensors.	178
9.3	Service domain monitoring by equipping containerized services with a forwarder for basic logging.	179
9.4	Elements of the operational context.	182

9.5	Schematic comparison of distributed versus centralized assurance of the ODD of automated driving functions.	183
9.6	Cybersecurity information management in the vehicle: centralized buffering and filtering.	186
9.7	Secure channel for transmission of cybersecurity monitoring data to the VSOC.	187
9.8	Integration of collected in-vehicle information and documents in the Knowledge Graph.	191
9.9	Overview of the ASCOT data model views.	193
9.10	Color code, connector legend and respective sections.	194
9.11	Development view.	195
9.12	Vehicle shadow view.	198
9.13	Analysis and handling view.	202
9.14	Threat information view.	203
10.1	Overview of the immediate response approach.	207
10.2	Categorization of response actions.	209
10.3	Workflow to activate pre-homologated responses over-the-air.	212
10.4	Interaction of response controller and course of action service.	214
11.1	Data flow diagram of the adaptive headlight system item.	219
11.2	The miniature vehicle.	220
11.3	Simulated and miniature vehicle coupled within environment simulation.	222
11.4	Comparison of real and simulated KIT campus.	223
11.5	Attack tree formalization of the attack path analysis for the exemplary item.	227
11.6	Attack tree with potential observables per attack step.	232
11.7	Activity diagram of the filter buffer in the CIM implementation.	237
11.8	Knowledge Graph after initialization with development documents.	243
11.9	Implementation of TARA and relationship to the threat information view in the KG.	245
11.10	Implementation of the development view in the KG.	246
11.11	Information collected from vehicle (C): Containerized software instances linked to the respective hosts and components.	249
11.12	Technical view of the implemented attack path in the miniature vehicle's E/E architecture.	250

11.13	Sequence diagram for attack step 1: initial access and persistence	250
11.14	Sequence diagram for attack step 2: command and control and execution	251
11.15	Sequence diagram for attack step 3: impact	253
11.16	Critical alerts produced by the HIDS on the implemented attack path.	254
11.17	Scene actions in Neo4j Desktop to analyze alerts.	255
11.18	Used data for initial analysis of the potential attack.	256
11.19	Environmental conditions of the vehicle during the alerts.	257
11.20	Incident response management application.	258
11.21	Additional information collected after increasing the log level.	259
11.22	Confirmed problem with the headlight assistant using its status information.	260
A.1	Terms used in the TARA and their relationship.	273
A.2	Terminology relationships.	274
A.3	Overview of the tactics and techniques in the ASCOT threat catalog.	282
A.4	List of observable information that were mapped to techniques.	282
A.5	Full ASCOT data model, combining all layers.	295
B.1	Data flow diagram of the keyless access item.	298
B.2	Attack tree for the keyless access item.	299
C.1	The layout of a CAN frame, numbers indicate the field length in <i>bit</i>	308
C.2	Elements of a SOA.	309

List of Tables

2.1	STRIDE model.	44
2.2	Parameters of attack feasibility.	48
2.3	Calculation of attack feasibility rating.	48
2.4	Calculation of impact rating.	48
2.5	Matrix to calculate risk value.	49
4.1	The monitoring domains of CMS, their action, and approximate categorization in the OSI model.	78
4.2	Incident response operation methodology in enterprise IT security standards and guidelines.	96
5.1	Comparison of database technologies.	109
5.2	Information obtained from data sources for monitoring, origin of data source, and classification in four views.	113
5.3	Comparison of the covered information of different data models. . .	117
7.1	Work products of the Preparation phase.	135
8.1	Monitoring goal matrix.	157
9.1	Information sources for holistic monitoring and their contribution to the four views.	168
11.1	Risk value calculation for identified attack paths.	230
11.2	Identified monitoring goals for the adaptive headlight system item. .	231
11.3	Mapping of techniques to cybersecurity monitoring sensors for the example attack path.	233
11.4	Resulting monitoring requirements and allocation to components in the exemplary architecture.	235
A.1	Terminology comparison of different standards and reports.	275
A.2	Terminology comparison of different standards and reports.	279
A.3	Chosen terminology definitions for this thesis.	280
A.4	Techniques and tactics	283

A.5	Observables and reference to corresponding data sources in MITRE ATT&CK®.	290
A.6	Cybersecurity monitoring sensors, domain and common potential observables.	292
A.7	Overview of classes across views of the ASCOT data model.	292
A.8	Response actions and potential actuators.	296
B.1	Attack steps in the Keyless access TARA.	300
B.2	Attack feasibility rating and risk value per attack path for keyless access item.	301
B.3	Selection of monitoring measures for keyless access item.	302
C.1	Common automotive and security protocols within the ISO/OSI model.	314

Bibliography

Own publications

- [GPS20] Daniel Grimm, Felix Pistorius, and Eric Sax. “Network Security Monitoring in Automotive Domain”. In: *Advances in Information and Communication*. Ed. by Kohei Arai, Supriya Kapoor, and Rahul Bhatia. Vol. 1129. Advances in Intelligent Systems and Computing. Cham: Springer International Publishing, 2020, pp. 782–799. ISBN: 978-3-030-39444-8. DOI: 10.1007/978-3-030-39445-5_57.
- [Gri+20] Daniel Grimm, Simon Leiner, Martin Sommer, Felix Pistorius, and Eric Sax. “Flow-based Aggregation of CAN Frames with Compressed Payload”. In: *2020 IEEE International Conference on Smart Computing (SMARTCOMP)*. 2020, pp. 202–207. DOI: 10.1109/SMARTCOMP50058.2020.00046.
- [Gri+23a] Daniel Grimm, Moritz Zink, Marc Schindewolf, and Eric Sax. “Adaptive Cybersecurity Monitoring for Resilient Vehicular Architectures”. In: *2023 IEEE Vehicular Networking Conference (VNC)*. 2023, pp. 41–48. DOI: 10.1109/VNC57357.2023.10136321.
- [Gri+23b] Daniel Grimm, Aljoscha Lautenbach, Magnus Almgren, Tomas Olovsson, and Eric Sax. “Gap Analysis of ISO/SAE 21434 – Improving the Automotive Cybersecurity Engineering Life Cycle”. In: *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*. 2023, pp. 1904–1911.

- [Gri+24a] Daniel Grimm, Dávid Tollner, David Kraus, Árpád Török, Eric Sax, and Zsolt Szalay. “A numerical verification method for multi-class feed-forward neural networks”. In: *Expert Systems with Applications* 247 (2024), p. 123345. ISSN: 0957-4174. DOI: <https://doi.org/10.1016/j.eswa.2024.123345>. URL: <https://www.sciencedirect.com/science/article/pii/S0957417424002100>.
- [Gri+24b] Daniel Grimm, Marc Schindewolf, David Kraus, and Eric Sax. “Co-simulate no more: The CARLA V2X Sensor”. In: *2024 IEEE Intelligent Vehicles Symposium (IV)*. 2024, pp. 2429–2436. DOI: 10.1109/IV55156.2024.10588488.
- [Gri+24c] Daniel Grimm, Moritz Zink, Marc Schindewolf, and Eric Sax. “Cyber Situational Awareness in Vehicle Security Operations: Holistic Monitoring and a Data Model”. In: *2024 20th International Conference on Network and Service Management (CNSM)*. 2024. DOI: 10.23919/CNSM62983.2024.10814319.
- [GS22] Daniel Grimm and Eric Sax. “Context-aware vehicle and fleet security combining a Knowledge Graph and an object-oriented model”. In: *2022 International Conference on Connected Vehicle and Expo (ICCVE)*. 2022, pp. 1–8. DOI: 10.1109/ICCVE52871.2022.9742971.
- [GSS21] Daniel Grimm, Marco Stang, and Eric Sax. “Context-Aware Security for Vehicles and Fleets: A Survey”. In: *IEEE Access* 9 (2021), pp. 101809–101846. DOI: 10.1109/ACCESS.2021.3097146.
- [GSS23] Daniel Grimm, Marc Schindewolf, and Eric Sax. “Fleet in the Loop: An Open Source approach for design and test of resilient vehicle architectures”. In: *2023 IEEE 15th International Symposium on Autonomous Decentralized System (ISADS)*. 2023, pp. 1–8. DOI: 10.1109/ISADS56919.2023.10092108.

- [GWS18] Daniel Grimm, Marc Weber, and Eric Sax. “An Extended Hybrid Anomaly Detection System for Automotive Electronic Control Units Communicating via Ethernet - Efficient and Effective Analysis using a Specification- and Machine Learning-based Approach”. In: *VEHITS 2018*. Ed. by Markus Helfert and Oleg Gusikhin. Setúbal, Portugal: SCITEPRESS - Science and Technology Publications Lda, 2018. ISBN: 978-989-758-293-6.
- [Haa+25] Yuna Haas, Daniel Baumann, Nikola Lukezic, Daniel Grimm, Luca Seidel, Vitus Lüntzel, David Kraus, and Eric Sax. “Paving the Road for Automated Buses in Public Transit: Technologies and Challenges”. In: *2025 9th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*. 2025, pp. 1–6. doi: 10.1109/MT-ITS68460.2025.11223594.
- [Pis+19] Felix Pistorius, Daniel Grimm, Marcel Auer, and Eric Sax. “Time Series Classification of Automotive Test Drives Using an Interval Based Elastic Ensemble”. In: *International Conference on Time Series and Forecasting (ITISE 2019)*. Universidad de Granada, 2019.
- [Pis+20] Felix Pistorius, Daniel Grimm, Florian Erdösi, and Eric Sax. “Evaluation Matrix for Smart Machine-Learning Algorithm Choice”. In: *2020 1st International Conference on Big Data Analytics and Practices (IBDAP)*. 2020, pp. 1–6.
- [Pud+22] Andreas Puder, Marcel Rumez, Daniel Grimm, and Eric Sax. “Generic Patterns for Intrusion Detection Systems in Service-Oriented Automotive and Medical Architectures”. In: *Journal of Cybersecurity and Privacy* 2.3 (2022), pp. 731–749. ISSN: 2624-800X. doi: 10.3390/jcp2030037. URL: <https://www.mdpi.com/2624-800X/2/3/37>.
- [Rum+20] Marcel Rumez¹, Daniel Grimm¹, Reiner Kriesten, and Eric Sax. “An Overview of Automotive Service-Oriented Architectures and Implications for Security Countermeasures”. In: *IEEE Access*

- 8 (2020). ¹: Both authors contributed equally to this work., pp. 221852–221870. doi: 10 . 1109/ACCESS . 2020 . 3043070.
- [Sch+22] Marc Schindewolf, Daniel Grimm, Christian Lingor, and Eric Sax. “Toward a Resilient Automotive Service-Oriented Architecture by using Dynamic Orchestration”. In: *1st IEEE International Conference on Cognitive Mobility*. Budapest, Hungary, 2022.
- [Sch+23] Marc Schindewolf, Jan Willem Wittler, Thomas Kühn, Daniel Grimm, and Eric Sax. “A Model-Based Approach to Automotive Feature Development for Updates and Upgrades”. In: *2023 IEEE International Conference on Service-Oriented System Engineering (SOSE)*. 2023, pp. 19–26. doi: 10 . 1109/SOSE58276 . 2023 . 00009.
- [Sta+20] Marco Stang, Daniel Grimm, Moritz Gaiser, and Eric Sax. “Evaluation of Deep Reinforcement Learning Algorithms for Autonomous Driving”. In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. 2020, pp. 1576–1582.
- [Sto+20a] Simon Stock, Alain Bertemes, Marco Stang, Martin Böhme, Daniel Grimm, and Wilhelm Stork. “FEEDI - A Smart Wearable Foot-Band for Navigation and Guidance Using Haptic Feedback”. In: *Human Interaction, Emerging Technologies and Future Applications II*. Ed. by Tareq Ahram, Redha Taiar, Vincent Gremeaux-Bader, and Kamiar Aminian. Cham: Springer International Publishing, 2020, pp. 349–355. ISBN: 978-3-030-44267-5.
- [Sto+20b] Simon Claus Stock, Jürgen Becker, Daniel Grimm, Tim Hotfilter, Gabriela Molinar, Marco Stang, and Wilhelm Stork. “QUA³CK - A Machine Learning Development Process”. In: *PoS AISIS2019* (2020), p. 026. doi: 10 . 22323/1 . 372 . 0026.

- [Sto+21] Hannes Stoll, Daniel Grimm, Marc Schindewolf, Michel Brodatzki, and Eric Sax. “Dynamic Reconfiguration of Automotive Architectures Using a Novel Plug-and-Play Approach”. In: *2021 IEEE Intelligent Vehicles Symposium Workshops (IV Workshops)*. 2021, pp. 70–75. doi: 10.1109/IVWorkshops54471.2021.9669222.
- [Vet+20] Andreas Vetter, Philipp Obergfell, Houssein Guissouma, Daniel Grimm, Marcel Rumez, and Eric Sax. “Development Processes in Automotive Service-oriented Architectures”. In: *2020 9th Mediterranean Conference on Embedded Computing (MECO)*. IEEE, June 2020, pp. 1–7. doi: 10.1109/MECO49872.2020.9134175.
- [ZGS26] Moritz Zink, Daniel Grimm, and Eric Sax. “AURORA Networks: Auto-associative Universal Real-Time Outlier Risk Assessment Networks”. In: *Computer Safety, Reliability, and Security. SAFE-COMP 2025 Workshops*. Ed. by Martin Törngren, Barbara Gallina, Erwin Schoitsch, Elena Troubitsyna, and Friedemann Bitsch. Cham: Springer Nature Switzerland, 2026, pp. 499–510. ISBN: 978-3-032-02018-5.
- [Zin+26] Moritz Zink, Daniel Baumann, Luca Seidel, Daniel Grimm, and Eric Sax. “ARTEMIS – Adaptive Response and Tracking of External Model Inputs in Safety-Critical Systems”. In: *IEEE International Conference on Recent Advances in Systems Science and Engineering (RASSE 2025)*. 2026.

References

- [Adm22] National Highway Traffic Safety Administration. “Cybersecurity best practices for the safety of modern vehicles”. In: (Sept. 2022). URL: <https://www.nhtsa.gov/sites/nhtsa.gov/f>

- iles/2022-09/cybersecurity-best-practices-safety-modern-vehicles-2022-tag.pdf.
- [Ågr+22] S. Magnus Ågren et al. “Agile Beyond Teams and Feedback Beyond Software in Automotive Systems”. In: *IEEE Transactions on Engineering Management* 69.6 (2022), pp. 3459–3475. doi: 10.1109/TEM.2022.3146139.
- [AK+24] Mehrnoosh Askarpour, Sahar Kokaly, et al. “An Exploration of Agile Methods in the Automotive Industry: Benefits, Challenges and Opportunities”. In: *arXiv preprint arXiv:2409.12676* (2024).
- [Al+19] Omar Y. Al-Jarrah et al. “Intrusion Detection Systems for Intra-Vehicle Networks: A Review”. In: *IEEE Access* 7 (2019), pp. 21266–21289. doi: 10.1109/ACCESS.2019.2894183.
- [Alb+04] Christopher J. Alberts et al. *Defining Incident Management Processes for CSIRTs: A Work in Progress*. Ed. by Carnegie Mellon University - Software Engineering Institute. 2004.
- [Ali16] Rizwan Ali. “Cybersituational awareness for the NATO alliance”. In: *The Three Swords Magazine* 30 (July 2016), pp. 72–75.
- [Ana+20] Sofia Ananieva et al. “A conceptual model for unifying variability in space and time”. In: *Proc. of the 24th ACM Conference on Systems and Software Product Line: Volume A - Volume A*. SPLC ’20. Montreal, Quebec, Canada: ACM, 2020. doi: 10.1145/3382025.3414955.
- [Ani23] Zainab Anifowose. “Aligning Mitre attack framework with threat analysis and risk assesment (TARA) to support R155 compliance”. MA thesis. Luleå University of Technology, Department of Computer Science, Electrical and Space Engineering, 2023, p. 47.
- [Ant20] Vard Antinyan. “Revealing the Complexity of Automotive Software”. In: *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2020.

- Virtual Event, USA: Association for Computing Machinery, 2020, pp. 1525–1528. ISBN: 9781450370431. DOI: 10.1145/3368089.3417038. URL: <https://doi.org/10.1145/3368089.3417038>.
- [Ant23] Vard Antinyan. “Seven Lessons Learned From Automotive Software Supplier Collaborations”. In: *IEEE Software* 40.1 (2023), pp. 77–85. DOI: 10.1109/MS.2022.3151166.
- [Arm+24] Alessandro Armellin et al. “A Framework for the Deployment of Cybersecurity Monitoring Tools in the Industrial Environment”. In: *IT Professional* 26.4 (2024), pp. 62–70. DOI: 10.1109/MITP.2024.3396356.
- [Bar+23a] Vita Santa Barletta et al. “Automotive Knowledge Base for Supporting Vehicle-SOC Analysts”. In: *2023 IEEE International Conference on Metrology for eXtended Reality, Artificial Intelligence and Neural Engineering (MetroXRINE)*. 2023, pp. 960–965. DOI: 10.1109/MetroXRINE58569.2023.10405622.
- [Bar+23b] Vita Santa Barletta et al. “V-SOC4AS: A Vehicle-SOC for Improving Automotive Security”. In: *Algorithms* 16.2 (2023). ISSN: 1999-4893. DOI: 10.3390/a16020112. URL: <https://www.mdpi.com/1999-4893/16/2/112>.
- [Bar19] Andrea Barisani. “Insecure Boot”. In: *17th escar Europe*. Ruhr-Universität Bochum, Nov. 2019.
- [Bau+23] Thomas Bauer et al. “Reference Architectures for Automotive Software”. In: *Reference Architectures for Critical Domains : Industrial Uses and Impacts*. Ed. by Elisa Yumi Nakagawa and Pablo Oliveira Antonino. Cham: Springer International Publishing, 2023, pp. 73–111. ISBN: 978-3-031-16957-1. DOI: 10.1007/978-3-031-16957-1_5. URL: https://doi.org/10.1007/978-3-031-16957-1_5.

- [BB07] Andrea Barisani and Daniele Bianco. *Unusual Car Navigation Tricks: Injecting RDS-TMC Traffic Information Signals*. Vancouver, Apr. 2007. URL: <https://cansecwest.com/csw07/csw07-barisani-bianco.pdf>.
- [BDS19] Ondrej Burkacky, Johannes Deichmann, and Jan Paul Stein. *Automotive software and electronics 2030: Mapping the sector's future landscape*. Ed. by McKinsey & Company. July 2019. URL: <https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/mapping-the-automotive-software-and-electronics-landscape-through-2030> (visited on 08/12/2020).
- [Ber19] Chris Berg. *How to Develop a Mixed-critical AUTOSAR Adaptive ECU with Safety and Security by Design*. Webinar, May 2019. URL: <https://www.sysgo.com/news-events/news-articles/article/webinar-how-to-develop-a-mixed-critical-autosar-adaptive-ecu-with-safety-and-security-by-design>.
- [Bes+23] Maciej Besta et al. "Demystifying Graph Databases: Analysis and Taxonomy of Data Organization, System Designs, and Graph Queries". In: *ACM Comput. Surv.* 56.2 (Sept. 2023). ISSN: 0360-0300. DOI: 10.1145/3604932. URL: <https://doi.org/10.1145/3604932>.
- [Beu19] Werner Beutnagel. 2019. URL: <https://www.automotiveintel.eu/strategy/renault-volvo-und-hyundai-s-teigen-bei-upstream-security-ein-233.html> (visited on 02/04/2025).
- [BK21] Robin Bolz and Reiner Kriesten. "Automotive Vulnerability Disclosure: Stakeholders, Opportunities, Challenges". In: *Journal of Cybersecurity and Privacy* 1.2 (2021), pp. 274–288. ISSN: 2624-800X. DOI: 10.3390/jcp1020015. URL: <https://www.mdpi.com/2624-800X/1/2/15>.

- [Boo+23] Lisa Boor et al. *J.D. Power 2023 U.S. Mobility Confidence Index (MCI) Study*. Ed. by J.D. Power, MIT Advanced Vehicle Technology Consortium. Oct. 2023.
- [Bro+09] Manfred Broy et al. *Automotive architecture framework: Towards a holistic and standardised system architecture description*. Tech. rep. Technische Universität München, 2009.
- [Bro06] Manfred Broy. “Challenges in automotive software engineering”. In: *Proceedings of the 28th international conference on Software engineering*. Ed. by Leon J. Osterweil. New York, NY: ACM, 2006, p. 33. doi: 10.1145/1134285.1134292.
- [Bur+18] Ondrej Burkacky et al. *Rethinking car software and electronics architecture*. Ed. by McKinsey & Company. Feb. 2018. URL: <https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/rethinking-car-software-and-electronics-architecture> (visited on 08/12/2020).
- [Bur+20] Ondrej Burkacky et al. *Cybersecurity in automotive: Mastering the challenge*. Ed. by McKinsey & Company and GSA. Mar. 2020. URL: <https://www.gsaglobal.org/wp-content/uploads/2020/03/Cybersecurity-in-automotive-Mastering-the-challenge.pdf> (visited on 08/12/2020).
- [Bur+23] Ondrej Burkacky et al. *Getting ready for next-generation E/E architecture with zonal compute*. Ed. by McKinsey & Company and GSA. June 2023. URL: <https://www.mckinsey.com/industries/semiconductors/our-insights/getting-ready-for-next-generation-ee-architecture-with-zonal-compute/> (visited on 08/30/2023).
- [BUR06] Chris Buechler, Scott Ullrich, and Rubicon Communications LLC (Netgate). *pfSense*. Version 2.7.2. Oct. 14, 2006.

- [C2A24] C2A Security. *List of Discontinued Car Models due to UN Regulation No. 155 on Vehicle Cybersecurity*. May 2024. URL: <https://c2a-sec.com/list-of-discontinued-car-models-due-to-un-regulation-no-155-on-vehicle-cybersecurity/> (visited on 12/06/2024).
- [Cha+23] Thibault Charmet et al. “Overview of the Operational Design Domain Monitoring for Safe Intelligent Vehicle Navigation”. In: *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*. 2023, pp. 5363–5370. DOI: 10.1109/ITSC57777.2023.10421823.
- [Cha09] Robert N. Charette. *This Car Runs On Code*. Ed. by IEEE Computer Society. 2009. URL: <https://spectrum.ieee.org/transportation/systems/this-car-runs-on-code> (visited on 09/13/2019).
- [Che+11] Stephen Checkoway et al. “Comprehensive Experimental Analyses of Automotive Attack Surfaces”. In: *Proceedings of the 20th USENIX Conference on Security*. SEC’11. USA: USENIX Association, 2011, p. 6.
- [Chl+23] Sebastian Chlup et al. “THREATGET: Towards Automated Attack Tree Analysis for Automotive Cybersecurity”. In: *Information* 14.1 (2023). ISSN: 2078-2489. DOI: 10.3390/info14010014. URL: <https://www.mdpi.com/2078-2489/14/1/14>.
- [Chr+17] Paul F. Christiano et al. “Deep reinforcement learning from human preferences”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS’17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 4302–4310. ISBN: 9781510860964.
- [Clo09] Mike Cloppert. “Security intelligence: Attacking the cyber kill chain”. In: *SANS Computer Forensics* 26 (2009).
- [Col14] Michael Collins. *Network Security Through Data Analysis*. O’Reilly Media, Inc., Feb. 2014. ISBN: 9781449357900.

-
- [Con19] Continental. *Continental Vehicle Server Connects VW ID. Electric Vehicles*. Nov. 2019. URL: <https://www.continental.com/en/press/press-releases/2019-11-12-icas-vw/> (visited on 08/30/2023).
- [Coo08] Robert G. Cooper. “Perspective: The Stage-Gate® Idea-to-Launch Process—Update, What’s New, and NexGen Systems”. In: *Journal of Product Innovation Management* 25.3 (2008), pp. 213–232. doi: <https://doi.org/10.1111/j.1540-5885.2008.00296.x>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-5885.2008.00296.x>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-5885.2008.00296.x>.
- [Cos+22] Breno Costa et al. “Orchestration in Fog Computing: A Comprehensive Survey”. In: *ACM Comput. Surv.* 55.2 (Jan. 2022). ISSN: 0360-0300. doi: 10.1145/3486221. URL: <https://doi.org/10.1145/3486221>.
- [Cur25] Sam Curry. *Hacking Subaru: Tracking and Controlling Cars via the STARLINK Admin Panel*. Jan. 2025. URL: <https://samcurry.net/hacking-subaru> (visited on 04/01/2026).
- [CZC18] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. “A comprehensive survey of graph embedding: Problems, techniques, and applications”. In: *IEEE transactions on knowledge and data engineering* 30.9 (2018), pp. 1616–1637.
- [DA+00] Anind K Dey, Gregory D Abowd, et al. “Towards a better understanding of context and context-awareness”. In: *CHI 2000 workshop on the what, who, where, when, and how of context-awareness*. Vol. 4. 2000, pp. 1–6.
- [Dag+17] Tsvika Dagan et al. “Vehicle Safe-Mode, Limp-Mode in the Service of Cyber Security”. In: *15th escar Europe*. 2017.

- [De +24] Marco De Vincenzi et al. “A Systematic Review on Security Attacks and Countermeasures in Automotive Ethernet”. In: *ACM Comput. Surv.* 56.6 (Jan. 2024). ISSN: 0360-0300. DOI: 10.1145/3637059.
- [Dei+19] Johannes Deichmann et al. *The race for cybersecurity: Protecting the connected car in the era of new regulation: The car industry’s digital transformation exposes new cybersecurity threats. Learn what OEMs can do to protect their cars and customers from hackers.* Ed. by McKinsey & Company. Oct. 2019. URL: <https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/the-race-for-cybersecurity-protecting-the-connected-car-in-the-era-of-new-regulation> (visited on 08/12/2020).
- [Dix23] Richard Dixon. *Be ready for the coming shift in automotive computing power.* S&P Global Mobility. Apr. 26, 2023. URL: <https://www.spglobal.com/mobility/en/research-analysis/be-ready-for-the-coming-shift-in-automotive-computing-power.html> (visited on 01/02/2025).
- [Dob+23] Jürgen Dobaj et al. “Towards a security-driven automotive development lifecycle”. In: *Journal of Software: Evolution and Process* 35.8 (2023), e2407. DOI: <https://doi.org/10.1002/smr.2407>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/smr.2407>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.2407>.
- [Dos+17] Alexey Dosovitskiy et al. *CARLA: An Open Urban Driving Simulator.* 2017. URL: <https://arxiv.org/pdf/1711.03938>.

-
- [DSK21] Jürgen Dürrwang, Florian Sommer, and Reiner Kriesten. “Automation in automotive security by using attacker privileges”. In: *19th escar Europe*. 2021. doi: 10.13154/294-8357.
- [Dup+21] Sébastien Dupont et al. “Incremental Common Criteria Certification Processes using DevSecOps Practices”. In: *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. 2021, pp. 12–23. doi: 10.1109/EuroSPW54576.2021.00009.
- [DW99] Wolfgang Dröschel and Manuela Wiemers, eds. *Der Standard für die Entwicklung von IT-Systemen mit Anleitung für den Praxiseinsatz*. Berlin, Boston: Oldenbourg Wissenschaftsverlag, 1999. ISBN: 9783486800265. doi: doi: 10.1515/9783486800265.
- [Eck+25] Magnus Wiik Eckhoff et al. “Experimenting with Neurosymbolic Artificial Intelligence for Defending Against Cyber Attacks”. In: *Neurosymbolic Artificial Intelligence 1* (2025), p. 29498732251377352. doi: 10.1177/29498732251377352. URL: <https://doi.org/10.1177/29498732251377352>.
- [Eic23] Jörn Eichler. *Maturing Automotive Security Engineering*. 21st es-car Europe. Hamburg, Nov. 2023.
- [EJ22] Christof Ebert and Jerome John. “Practical Cybersecurity with ISO 21434”. In: *ATZelectronics worldwide* 17 (Mar. 2022). doi: 10.1007/s38314-021-0741-5.
- [End88] Mica R Endsley. “Design and evaluation for situation awareness enhancement”. In: *Proc. of the Human Factors Society annual meeting*. Vol. 32, 2. Sage Publications Sage CA: Los Angeles, CA, 1988, pp. 97–101. doi: 10.1177/154193128803200221.
- [Erz+22] Jannis Erz et al. “Towards an Ontology That Reconciles the Operational Design Domain, Scenario-based Testing, and Automated

- Vehicle Architectures”. In: *2022 IEEE International Systems Conference (SysCon)*. 2022, pp. 1–8. doi: 10.1109/SysCon53536.2022.9773840.
- [Eur21] UN Economic Commission for Europe (UNECE). *Proposals for the Interpretation Documents for UN Regulation No. 155 (Cyber security and cyber security management system)*. Mar. 2021. URL: https://unece.org/sites/default/files/2021-02/ECE-TRANS-WP29-2021-059e_0.pdf (visited on 12/18/2024).
- [FC17] James B. Fraley and James Cannady. “The promise of machine learning in cybersecurity”. In: *SoutheastCon 2017*. 2017, pp. 1–6. doi: 10.1109/SECON.2017.7925283.
- [FCM21] Marta Fuentes-García, José Camacho, and Gabriel Maciá-Fernández. “Present and future of network security monitoring”. In: *IEEE Access* 9 (2021), pp. 112744–112760.
- [Fis23] Justin Fischer. *Over-the-Air (OTA) Updates: How Automakers Are Betting on Software to Keep Up With the Pace of Innovation*. Oct. 2023. URL: <https://caredge.com/guides/ota-updates-for-cars> (visited on 06/28/2024).
- [För+19] David Förster et al. “Safety goals in vehicle security analyses - a method to assess malicious attacks with safety impact”. In: *17th escar Europe*. 2019. doi: 10.13154/294-6657.
- [Fra+22] Ulrik Franke et al. “Cyber situational awareness issues and challenges”. In: *Cybersecurity and Cognitive Science*. Ed. by Ahmed A. Moustafa. Academic Press, 2022. Chap. 10, pp. 235–265. doi: 10.1016/B978-0-323-90570-1.00015-2.
- [FT] Eclipse Foundation and The German Aerospace Center (DLR) Institute for Transportation Systems. *Eclipse ADORe*. URL: <https://eclipse-adore.github.io/adore/> (visited on 10/27/2025).

- [GB98] Erik Guttman and Nevil Brownlee. *Expectations for Computer Security Incident Response*. Internet Request for Comments. June 1998. doi: 10.17487/RFC2350. URL: <https://www.rfc-editor.org/info/rfc2350>.
- [GDF21] Josep Soler Garrido, Dominik Dold, and Johannes Frank. “Machine learning on knowledge graphs for context-aware security monitoring”. In: *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*. 2021, pp. 55–60. doi: 10.1109/CSR51186.2021.9527927.
- [Ger09] R. Gerhards. *The Syslog Protocol*. RFC 5424. RFC Editor, Mar. 2009. URL: <https://www.rfc-editor.org/rfc/rfc5424>.
- [GH24] Eyosias Dawit Guyo and Timo Hartmann. “Evaluating the efficiency and performance of data persistent systems in managing building and environmental Data: A comparative study”. In: *Advanced Engineering Informatics* 62 (2024), p. 102582. ISSN: 1474-0346. doi: <https://doi.org/10.1016/j.aei.2024.102582>. URL: <https://www.sciencedirect.com/science/article/pii/S1474034624002301>.
- [Gol+21] Fay Goldstein et al. “Monitoring Automotive Cyber Risks Throughout The Deep and Dark Web”. In: *19th escar Europe*. 2021.
- [Gör+22] Lorenz Görne et al. “Smart data preprocessing method for remote vehicle diagnostics to increase data compression efficiency”. In: *Automotive and Engine Technology* (2022), pp. 1–10. doi: 10.1007/s41104-022-00113-9.
- [Gra+19] Joachim Graf et al. “The Challenge of being safe and secure - designing process interfaces between safety and security”. In: *17th escar Europe*. 2019. doi: 10.13154/294-6658.

- [Gre+22] Simon Greiner et al. “A supplier’s perspective on threat analysis and risk assessment according to ISO/SAE 21434”. In: *20th escar Europe*. 2022.
- [Gre15] Andy Greenberg. July 2015. URL: <https://www.wired.com/2015/07/hackers-remotely-kill-jep-highway> (visited on 07/26/2024).
- [Gro23] Automotive Security Research Group. *Threat Catalog*. Mar. 2023. URL: <https://garage.asrg.io/threat-catalog/> (visited on 04/03/2024).
- [Gua15] The Guardian, ed. July 2015. URL: <https://www.theguardian.com/business/2015/jul/24/fiat-chrysler-recall-jep-hacking> (visited on 07/26/2024).
- [Gui+18] Housseem Guissouma et al. “An Empirical Study on the Current and Future Challenges of Automotive Software Release and Configuration Management”. In: *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. 2018, pp. 298–305. DOI: 10.1109/SEAA.2018.00056.
- [Gui+22] Housseem Guissouma et al. “Lifecycle Management of Automotive Safety-Critical Over the Air Updates: A Systems Approach”. In: *IEEE Access* 10 (2022), pp. 57696–57717. DOI: 10.1109/ACCESS.2022.3176879.
- [GZS23] Housseem Guissouma, Moritz Zink, and Eric Sax. “Continuous Safety Assessment of Updated Supervised Learning Models in Shadow Mode”. In: *2023 IEEE 20th International Conference on Software Architecture Companion (ICSA-C)*. 2023, pp. 301–308. DOI: 10.1109/ICSA-C57050.2023.00069.
- [Häc+23] Timo Häckel et al. *A Multilayered Security Infrastructure for Connected Vehicles – First Lessons from the Field*. BROAD workshop at the 2022 IEEE Intelligent Vehicles Symposium (IV) in Aachen, Germany. 2023. arXiv: 2310.10336.

- [Hal+21] Lavdim Halilaj et al. “A Knowledge Graph-Based Approach for Situation Comprehension in Driving Scenarios”. In: *The Semantic Web*. Ed. by Ruben Verborgh et al. Cham: Springer International Publishing, 2021, pp. 699–716. doi: 10.1007/978-3-030-77385-4_42.
- [Ham20] Mohammad Hamad. “A multilayer secure framework for vehicular systems”. PhD thesis. Technische Universität Braunschweig, Braunschweig, Germany, 2020.
- [HB12] Florian Hartwich and Robert Bosch. “CAN with Flexible Data-Rate”. In: *Proceedings of the 13th international CAN Conference*. Hambach Castle, Germany, 2012.
- [HB20] Florian Hartwich and Robert Bosch. “Introducing CAN XL into CAN Networks”. In: *Proceedings of the 17th international CAN Conference*. Baden-Baden, 2020.
- [HBH23] Jenny Hofbauer, Kevin Klaus Gomez Buquerin, and Hans-Joachim Hof. “From SOC to VSOC”. In: *21st escar Europe*. Transferring key requirements for efficient vehicle security operations. 2023. doi: 10.13154/294-10389.
- [HC20] Guy Harpak and Yuankai Chen. *Mercedes-Benz and 360 Group: Defending a Luxury Fleet with the Community*. RSA Conference. San Francisco, Feb. 2020. URL: <https://www.rsaconference.com/library/presentation/usa/2020/mercedes-benz-and-360-group-defending-a-luxury-fleet-with-the-community> (visited on 01/24/2024).
- [hei19] heise magazine. “Continental Auto-Server für VW”. In: *c’t 26* (2019), p. 49. URL: <https://www.heise.de/select/ct/2019/26/1576519121670761> (visited on 12/06/2024).
- [Hen+23] Henle, Jacqueline et al. “Concept for an Approval-Focused Over-The-Air Update Development Process”. In: *23rd Stuttgart International Symposium*. SAE International, June 2023. doi: <https://www.sae.org/publications/abstracts/2023-01-0190/>

- [//doi.org/10.4271/2023-01-1224](https://doi.org/10.4271/2023-01-1224). URL: <https://doi.org/10.4271/2023-01-1224>.
- [Her+24] Teresina Herb et al. *Automotive Cyber Maturity Report 2023*. ETAS GmbH. July 2024.
- [HG18] Gunnar Harde and Jürgen Großmann. “Car Security Incident Response - Project Report”. In: *Automotive Quality Institute* (June 2018). URL: https://aqigmbh.de/wp-content/uploads/2022/04/20190207_FinalReport_CarSIR.pdf.
- [HKD08a] Tobias Hoppe, Stefan Kiltz, and Jana Dittmann. “Adaptive Dynamic Reaction to Automotive IT Security Incidents Using Multimedia Car Environment”. In: *2008 The Fourth International Conference on Information Assurance and Security*. 2008, pp. 295–298. DOI: 10.1109/IAS.2008.45.
- [HKD08b] Tobias Hoppe, Stefan Kiltz, and Jana Dittmann. “Security Threats to Automotive CAN Networks – Practical Examples and Selected Short-Term Countermeasures”. In: *Computer Safety, Reliability, and Security 27th International Conference, SAFECOMP 2008 Newcastle upon Tyne, UK, September 22-25, 2008 Proceedings*. Ed. by Michael D. Harrison and Mark-Alexander Sujan. Lecture Notes in Computer Science. Berlin: Springer, 2008, pp. 235–248. ISBN: 978-3-540-87698-4.
- [Hug+19] Vincent Hugot et al. *oMAC : Open Model for Automotive Cyber-security*. Ruhr-Universität Bochum, 2019. DOI: 10.13154/294-6674.
- [Hus+22] Martin Husák et al. “CRUSOE: A toolset for cyber situational awareness and decision support in incident handling”. In: *Computers & Security* 115 (2022). DOI: <https://doi.org/10.1016/j.cose.2022.102609>.

- [HWH19] James Halvorsen, Jesse Waite, and Adam Hahn. “Evaluating the Observability of Network Security Monitoring Strategies With TOMATO”. In: *IEEE Access* 7 (2019), pp. 108304–108315. doi: 10.1109/ACCESS.2019.2933415.
- [HYL20] Guy Harpak, Minrui Yan, and Jiahao Li. *Security Research on Mercedes-Benz: From Hardware to Car Control*. Black Hat USA Briefings. Virtual Event, Aug. 2020. URL: <https://i.blackhat.com/USA-20/Thursday/us-20-Yan-Security-Research-On-Mercedes-Benz-From-Hardware-To-Car-Control.pdf> (visited on 07/22/2024).
- [Ill+24] Márton Illés et al. “Effective automotive vulnerability management with TARA based prioritization”. In: *22nd escar Europe*. 2024.
- [ite23] itemis AG. *openXSAM - Towards a common Security Analysis Exchange Format for ISO/SAE 21434 and UN Reg. 155*. June 2023. URL: <https://openxsam.io/wp-content/uploads/2023/06/openXSAM-Towards-a-common-Security-Analysis-Exchange-Format.pdf> (visited on 01/29/2024).
- [Jia+20] Jingxuan Jiang et al. “Road context-aware intrusion detection system for autonomous cars”. In: *Information and Communications Security: 21st International Conference, ICICS 2019, Beijing, China, December 15–17, 2019, Revised Selected Papers 21*. Springer. 2020, pp. 124–142.
- [JPD21] Bret Jordan, Rich Piazza, and Trey Darley, eds. *STIX Version 2.1*. OASIS Standard, June 2021. URL: <https://docs.oasis-open.org/cti/stix/v2.1/stix-v2.1.html>.
- [Jun22] Ramona Jung. “Cyber Security for Future Zonal E/E Architectures”. In: *20th escar Europe*. Ruhr-Universität Bochum, 2022.

- [KA21] Alexander Kiening and Daniel Angermeier. “TRADE - Threat and Risk Assessment for Automotive Distributed Engineering”. In: *19th escar Europe*. 2021. doi: 10.13154/294-8355.
- [Kai+23] Florian Klaus Kaiser et al. “Attack Hypotheses Generation Based on Threat Intelligence Knowledge Graph”. In: *IEEE Transactions on Dependable and Secure Computing* 20.6 (2023), pp. 4793–4809. doi: 10.1109/TDSC.2022.3233703.
- [KBH25] Isaiah J. King, Benjamin Bowman, and H. Howie Huang. “Automated Cyber Defense with Generalizable Graph-Based Reinforcement Learning Agents”. In: *arXiv preprint arXiv:2509.16151* (2025). arXiv: 2509.16151 [cs.LG]. URL: <https://arxiv.org/abs/2509.16151>.
- [KBS05] Dirk Krafzig, Karl Banke, and Dirk Slama. *Enterprise SOA: service-oriented architecture best practices*. Prentice Hall Professional, 2005.
- [KEK21] Kabul Kurniawan, Andreas Ekelhart, and Elmar Kiesling. “An ATT&CK-KG for Linking Cybersecurity Attacks to Adversary Tactics and Techniques”. In: *International Semantic Web Conference (ISWC) - Posters and Demos*. Oct. 2021. URL: <https://ceur-ws.org/Vol-2980/paper363.pdf>.
- [Ken21] Dennis Kengo Oka. *Building Secure Cars : Assuring the Automotive Software Development Lifecycle*. Wiley, 2021. ISBN: 9781119710745.
- [KG99] Loren Kohnfelder and Praerit Garg. “The threats to our products”. In: *Microsoft Interface, Microsoft Corporation* 33 (1999).
- [KHK21] Andrew Benjamin Koerner, Björn Hendriks, and Michael Kürschner. “Selective software updates with in situ monitoring of non-homogeneous automotive electronic control units”. In: *ACIMobility Summit 2021*. ACIMobility, Sept. 2021. URL: <https://elib.dlr.de/143270/>.

- [Kie+19] Elmar Kiesling et al. “The SEPSES Knowledge Graph: An Integrated Resource for Cybersecurity”. In: *The Semantic Web – ISWC 2019*. Ed. by Chiara Ghidini et al. Cham: Springer International Publishing, 2019, pp. 198–214. doi: 10.1007/978-3-030-30796-7_13.
- [KKE24] Kabul Kurniawan, Elmar Kiesling, and Andreas Ekelhart. “CyKG-RAG: Towards Knowledge-Graph Enhanced Retrieval Augmented Generation for Cybersecurity”. In: *Proceedings of the International Workshop on Retrieval-Augmented Generation Enabled by Knowledge Graphs (RAGE-KG 2024)*. Vol. 3950. CEUR Workshop Proceedings. Baltimore, USA: CEUR-WS.org, 2024, pp. 51–64. URL: <https://ceur-ws.org/Vol-3950/paper1.pdf>.
- [KM22] Maria Korolov and Lysa Myers. *What is the cyber kill chain? A model for tracing cyberattacks*. Apr. 2022. URL: <https://www.csoonline.com/article/539916/what-is-the-cyber-kill-chain-a-model-for-tracing-cyberattacks.html> (visited on 04/02/2024).
- [Kni+22] Christoph Knieke et al. “Managed Evolution of Automotive Software Product Line Architectures: A Systematic Literature Study”. In: *Electronics* 11.12 (2022).
- [Kom+18] Jana Komárková et al. “CRUSOE: Data Model for Cyber Situational Awareness”. In: *Proc. of the 13th International Conference on Availability, Reliability and Security. ARES ’18*. Hamburg, Germany: ACM, 2018. doi: 10.1145/3230833.3232798.
- [Kos+10] Karl Koscher et al. “Experimental Security Analysis of a Modern Automobile”. In: *IEEE Symposium on Security and Privacy (SP), 2010*. Piscataway, NJ: IEEE, 2010, pp. 447–462. ISBN: 978-1-4244-6894-2. doi: 10.1109/SP.2010.34.

- [KPZ23] Kathryn Knerler, Ingrid Parker, and Carson Zimmerman. *11 Strategies of a World-Class Cybersecurity Operations Center*. MITRE, 2023.
- [KŞV+22] Jonas Krog, Tarık Şahin, Thomas Vietor, et al. “Towards a systems engineering methodology for architecture development of vehicle concepts”. In: *DS 118: Proceedings of NordDesign 2022, Copenhagen, Denmark, 16th-18th August 2022* (2022), pp. 1–12.
- [KTK23] Shahidullah Kaiser, Ali şaman Tosun, and Turgay Korkmaz. “Benchmarking Container Technologies on ARM-Based Edge Devices”. In: *IEEE Access* 11 (2023), pp. 107331–107347. doi: 10.1109/ACCESS.2023.3321274.
- [Kuk+22] Nemanja Kukulicic et al. “Automotive Service-oriented Architectures: a Systematic Mapping Study”. In: *2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. 2022, pp. 459–466. doi: 10.1109/SEAA56994.2022.00077.
- [Kur+22] Kabul Kurniawan et al. “KRYSTAL: Knowledge graph-based framework for tactical attack discovery in audit data”. In: *Computers & Security* 121 (2022), p. 102828. issn: 0167-4048. doi: <https://doi.org/10.1016/j.cose.2022.102828>. URL: <https://www.sciencedirect.com/science/article/pii/S016740482200222X>.
- [LAO21] Aljoscha Lautenbach, Magnus Almgren, and Tomas Olovsson. “Proposing HEAVENS 2.0 – an automotive risk assessment model”. In: *Proc. of the 5th ACM Computer Science in Cars Symposium*. CSCS ’21. Ingolstadt, Germany: ACM, 2021. doi: 10.1145/3488904.3493378.
- [LHS22] Jan Lauinger, Mohammad Hamad, and Sebastian Steinhorst. “Toward a Multi-Layer Intrusion Response System for Connected Vehicles”. In: *2022 IEEE 96th Vehicular Technology Conference*

- (VTC2022-Fall). 2022, pp. 1–2. DOI: 10.1109/VTC2022-Fall.1157202. 2022.10012711.
- [LI16] Aljoscha Lautenbach and Mafijul Islam. “Deliverable D2: Security models”. In: *HEaling Vulnerabilities to ENhance Software Security and Safety (HEAVENS) project* (Mar. 2016).
- [Lie+24] Thomas Liedtke et al. “Consistency for More Than One TARA and Security Element Out of Context Experiences”. In: *Systems, Software and Services Process Improvement*. Ed. by Murat Yilmaz et al. Cham: Springer Nature Switzerland, 2024, pp. 315–336. ISBN: 978-3-031-71139-8.
- [Lie21] Thomas Liedtke. “Risk Assessment According to the ISO/SAE 21434:2021 – Experiences, Help and Pitfalls”. In: *19th escar Europe*. 2021.
- [LM23] Brooke Lampe and Weizhi Meng. “Intrusion Detection in the Automotive Domain: A Comprehensive Review”. In: *IEEE Communications Surveys & Tutorials* 25.4 (2023), pp. 2356–2426. DOI: 10.1109/COMST.2023.3309864.
- [LN24] Gordon Lyon and the Nmap-Developer Team. Version 7.95, Initial release: 1997. 2024. URL: <https://nmap.org/> (visited on 10/24/2024).
- [Lo+22] Wai Weng Lo et al. “E-GraphSAGE: A Graph Neural Network based Intrusion Detection System for IoT”. In: *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*. 2022, pp. 1–9. DOI: 10.1109/NOMS54207.2022.9789878.
- [Loe+25] Johannes F. Loevenich et al. “Automating Cyber Threat Intelligence and Attack Chain Generation using Cyber Security Knowledge Graphs and Large Language Models”. In: *2025 International Conference on Military Communication and Information Systems (ICMCIS)*. 2025, pp. 1–10. DOI: 10.1109/ICMCIS64378.2025.11047951.

- [LSS19] Falk Langer, Fabian Schüppel, and Lukas Stahlbock. “Establishing an Automotive Cyber Defense Center”. en. In: *17th escar Europe*. Ruhr-Universität Bochum, 2019. doi: 10.13154/294-6652.
- [LSS20] Falk Langer, Fabian Schüppel, and Lukas Stahlbock. “Incident response for vehicular systems - more than online updates”. In: *18th escar Europe*. Ruhr-Universität Bochum, 2020.
- [Lue+22] Juergen Luetttin et al. “A Survey on Knowledge Graph-Based Methods for Automated Driving”. In: *Knowledge Graphs and Semantic Web*. Ed. by Boris Villazón-Terrazas et al. Cham: Springer International Publishing, 2022, pp. 16–31. doi: 10.1007/978-3-031-21422-6_2.
- [Ma+24] Yuke Ma et al. “The Advancement of Knowledge Graphs in Cybersecurity: A Comprehensive Overview”. In: *Computational and Experimental Simulations in Engineering*. Ed. by Shaofan Li. Cham: Springer International Publishing, 2024, pp. 65–103.
- [Mad+23] Ralph Mader et al. “Signal/Service Translation-Where is the place to be?” In: *AmE 2023—Automotive meets Electronics; 14. GMM Symposium*. VDE. 2023, pp. 22–27.
- [Mat19] Kirsten Matheus. *Evolution of Ethernet-based Automotive Networks: Faster and Cheaper*. 8th Ethernet & IP @ AUTOMOTIVE TECHNOLOGY. London, Oct. 2019. URL: <https://standards.ieee.org/wp-content/uploads/import/documents/other/d1-03-matheus-evolution-of-ethernet-based-automotive-networks.pdf> (visited on 12/16/2024).
- [Mat24] MathWorks®. *System Composer*. 2024. URL: <https://de.mathworks.com/products/system-composer.html>.
- [May+24] Kevin Mayer et al. “Vehicle Security Operations Center for Cooperative, Connected and Automated Mobility”. In: *SECURWARE 2024 : The Eighteenth International Conference on Emerging Security Information, Systems and Technologies*. Nov. 2024.

- [MBB18] Mario Maul, Gerhard Becker, and Ulrich Bernhard. “Service-oriented EE zone architecture key elements for new market segments”. In: *ATZelektronik worldwide* 13 (2018), pp. 36–41. doi: 10.1007/s38314-017-0092-4.
- [MBL08] Gaspar Modelo-Howard, Saurabh Bagchi, and Guy Lebanon. “Determining Placement of Intrusion Detectors for a Distributed Application through Bayesian Network Modeling”. In: *Recent Advances in Intrusion Detection*. Ed. by Richard Lippmann, Engin Kirda, and Ari Trachtenberg. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 271–290. ISBN: 978-3-540-87403-4.
- [McC+14] Charlie McCarthy et al. *Assessment of the information sharing and analysis center model*. Tech. rep. National Highway Traffic Safety Administration, 2014. URL: www.nhtsa.gov/sites/nhtsa.dot.gov/files/812076-assessinfosharingmodel.pdf.
- [Mey+20] Philipp Meyer et al. “Demo: A Security Infrastructure for Vehicular Information Using SDN, Intrusion Detection, and a Defense Center in the Cloud”. In: *2020 IEEE Vehicular Networking Conference (VNC)*. 2020, pp. 1–2. doi: 10.1109/VNC51378.2020.9318351.
- [MGF10] Michael Müter, Andre Groll, and Felix C. Freiling. “A structured approach to anomaly detection for in-vehicle networks”. In: *2010 Sixth International Conference on Information Assurance and Security*. IEEE, 2010, pp. 92–98. ISBN: 978-1-4244-7407-3. doi: 10.1109/ISIAS.2010.5604050.
- [MH21] Kirsten Matheus and Daniel Hopf. *Is a Ubiquitous Automotive Ethernet (still) the Right Goal?* Jan. 2021. URL: <https://www.elektroniknet.de/international/is-a-ubiquitous-automotive-ethernet-still-the-right-goal.182086.html> (visited on 08/30/2023).

- [Mic+24] Davide Micale et al. “A context-aware on-board intrusion detection system for smart vehicles”. In: *International Journal of Information Security* 23.3 (June 2024), pp. 2203–2223. ISSN: 1615-5270. DOI: 10.1007/s10207-024-00821-3. URL: <https://doi.org/10.1007/s10207-024-00821-3>.
- [Mil11] David Miller. *Security Information and Event Management (SIEM) Implementation*. New York: McGraw-Hill, 2011.
- [Min23a] Moritz Minzlaff. *Automotive Cyber Maturity Report 2023*. ETAS GmbH. July 2023.
- [Min23b] Moritz Minzlaff. *Automotive Cyber Maturity Survey 2023 – An industry on the move*. 10th escar USA. June 2023.
- [Mog+24] Felix Mogge et al. *Automotive Outlook 2040*. Ed. by Roland Berger GmbH. Oct. 2024. URL: <https://www.rolandberger.com/en/Insights/Global-Topics/Automotive-2040/> (visited on 01/09/2025).
- [MP21] Stefan Marksteiner and Peter Priller. “A Model-Driven Methodology for Automotive Cybersecurity Test Case Generation”. In: *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. 2021, pp. 129–135. DOI: 10.1109/EuroSPW54576.2021.00021.
- [MS25] Inoussa Mouiche and Sherif Saad. “TIJERE: A novel threat intelligence joint extraction model based on analyst expert knowledge”. In: *Knowledge-Based Systems* 329 (2025), p. 114346. ISSN: 0950-7051. DOI: <https://doi.org/10.1016/j.knosys.2025.114346>. URL: <https://www.sciencedirect.com/science/article/pii/S0950705125013851>.
- [MV14] Charlie Miller and Chris Valasek. “A survey of remote automotive attack surfaces”. In: *Black Hat USA 2014* (2014), p. 94.

- [Naj+19] Pejman Najafi et al. “MalRank: A Measure of Maliciousness in SIEM-Based Knowledge Graphs”. In: *Proceedings of the 35th Annual Computer Security Applications Conference*. AC-SAC '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 417–429. ISBN: 9781450376280. DOI: 10.1145/3359789.3359791.
- [Nan+26] Padmeswari Nandiya et al. *BRIDG-ICS: AI-Grounded Knowledge Graphs for Intelligent Threat Analytics in Industry 5.0 Cyber-Physical Systems*. 2026. arXiv: 2512.12112 [cs.CR]. URL: <https://arxiv.org/abs/2512.12112>.
- [Nav+15] Varun M. Navale et al. “(R)evolution of E/E Architectures”. In: *SAE International Journal of Passenger Cars - Electronic and Electrical Systems* 8.2 (2015), pp. 282–288. ISSN: 1946-4622. DOI: 10.4271/2015-01-0196.
- [NBT22] Ali Nouri, Christian Berger, and Fredrik Törner. “An Industrial Experience Report about Challenges from Continuous Monitoring, Improvement, and Deployment for Autonomous Driving Features”. In: *2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. 2022, pp. 358–365. DOI: 10.1109/SEAA56994.2022.00063.
- [NCM21] Pejman Najafi, Feng Cheng, and Christoph Meinel. “SIEMA: Bringing Advanced Analytics to Legacy Security Information and Event Management”. In: *Security and Privacy in Communication Networks*. Ed. by Joaquin Garcia-Alfaro et al. Cham: Springer International Publishing, 2021, pp. 25–43. ISBN: 978-3-030-90019-9.
- [NDB10] Nicolas Navet, Bertrand Delord, and Markus Baumeister. “Virtualization in Automotive Embedded Systems : an Outlook”. In: *Seminar at RTS Embedded Systems*. Mar. 2010.

- [NLO17] Nasser Nowdehi, Aljoscha Lautenbach, and Tomas Olovsson. “In-Vehicle CAN Message Authentication: An Evaluation Based on Industrial Criteria”. In: *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*. 2017, pp. 1–7. doi: 10.1109/VTCFall.2017.8288327.
- [Noe+16] Steven Noel et al. “CyGraph: Graph-Based Analytics and Visualization for Cybersecurity”. In: *Cognitive Computing: Theory and Applications*. Ed. by Venkat N. Gudivada et al. Vol. 35. Handbook of Statistics. Elsevier, 2016, pp. 117–167. doi: <https://doi.org/10.1016/bs.host.2016.07.001>.
- [Oes+20] Sean Oesch et al. “An Assessment of the Usability of Machine Learning Based Tools for the Security Operations Center”. In: *2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*. 2020, pp. 634–641. doi: 10.1109/iThings-GreenCom-CPSCom-SmartData-Cybermatics50389.2020.00111.
- [OHB19] Habeeb Olufowobi, Sena Hounsinnou, and Gedare Bloom. “Controller Area Network Intrusion Prevention System Leveraging Fault Recovery”. In: *Proceedings of the ACM Workshop on Cyber-Physical Systems Security & Privacy*. CPS-SPC’19. London, United Kingdom: Association for Computing Machinery, 2019, pp. 63–73. ISBN: 9781450368315. doi: 10.1145/3338499.3357360. url: <https://doi.org/10.1145/3338499.3357360>.
- [OK19] Samir Ouchani and Abdelaziz Khaled. “A Meta Language for Cyber-Physical Systems and Threats: Application on Autonomous Vehicle”. In: *2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA)*. 2019, pp. 1–8. doi: 10.1109/AICCSA47632.2019.9035273.

- [Olt18] Christian Olt. “Cybersecurity for Connected Cars: Pathways to a Security Operation Center for the Automotive Industry”. In: (2018). Ed. by T-Systems International GmbH. URL: <https://www.telekom.com/resource/blob/565704/527b25d7f2e3d1e5aa94afd0fc760721/d1-pathways-automotive-soc-data.pdf> (visited on 01/21/2024).
- [Onw18] Cyril Onwubiko. “CoCoa: An Ontology for Cybersecurity Operations Centre Analysis Process”. In: *2018 International Conference On Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA)*. 2018, pp. 1–8. doi: 10.1109/CyberSA.2018.8551486.
- [Osi07] Frans PB Osinga. *Science, strategy and war: The strategic theory of John Boyd*. Routledge, 2007.
- [Ouy+22] Long Ouyang et al. “Training language models to follow instructions with human feedback”. In: *Proceedings of the 36th International Conference on Neural Information Processing Systems*. NIPS ’22. New Orleans, LA, USA: Curran Associates Inc., 2022. ISBN: 9781713871088.
- [Per24] Persistence Market Research. *Automotive Cyber Security Market*. Jan. 2024. URL: <https://www.globenewswire.com/news-release/2024/01/12/2808450/0/en/Automotive-Cyber-Security-Market-is-projected-to-grow-US-26-492-6-Million-by-2033-at-an-25-8-CAGR-by-2033-Persistence-Market-Research.html> (visited on 02/06/2024).
- [Pią22] Piotr Piątek. “Incident Management Process Model for Automotive CyberSafety Systems Using the Business Process Model and Notation”. In: *2022 26th International Conference on Methods and Models in Automation and Robotics (MMAR)*. 2022, pp. 232–237. doi: 10.1109/MMAR55195.2022.9874288.

- [Pla+21] Christian Plappert et al. “Attack Surface Assessment for Cybersecurity Engineering in the Automotive Domain”. In: *2021 29th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*. 2021, pp. 266–275. doi: 10.1109/PDP52278.2021.00050.
- [Poh+12] Klaus Pohl et al. *Model-based engineering of embedded systems: The SPES 2020 methodology*. Springer, 2012. doi: <https://doi.org/10.1007/978-3-642-34614-9>.
- [Poh+16] Klaus Pohl et al. *Advanced Model-Based Engineering of Embedded Systems: Extensions of the SPES 2020 Methodology*. Springer, 2016. doi: <https://doi.org/10.1007/978-3-319-48003-9>.
- [Pol24] Polaris Market Research. *Automotive Ethernet Market Share, Size, Trends, Industry Analysis Report*. Mar. 2024. URL: <https://www.polarismarketresearch.com/industry-analysis/automotive-ethernet-market> (visited on 01/02/2025).
- [PST22] Zsombor Pethő, Zsolt Szalay, and Árpád Török. “Safety risk focused analysis of V2V communication especially considering cyberattack sensitive network performance and vehicle dynamics factors”. In: *Vehicular Communications* 37 (2022), p. 100514. ISSN: 2214-2096. doi: <https://doi.org/10.1016/j.vehcom.2022.100514>. URL: <https://www.sciencedirect.com/science/article/pii/S2214209622000614>.
- [PSZ17] Mert D. Pesé, Karsten Schmidt, and Harald Zweck. “Hardware-Software Co-Design of an Automotive Embedded Firewall”. In: *SAE Technical Paper* (2017). doi: 10.4271/2017-01-1659. URL: <https://www.sae.org/content/2017-01-1659/>.

-
- [Ran+20] Konstantinos Rantos et al. “Interoperability Challenges in the Cybersecurity Information Sharing Ecosystem”. In: *Computers* 9.1 (2020). issn: 2073-431X. doi: 10.3390/computers9010018.
- [RHS22] Langley Rock, Stefan Hagen, and Thomas Schmidt, eds. *Common Security Advisory Framework Version 2.0*. Latest stage: <https://docs.oasis-open.org/csaf/csaf/v2.0/csaf-v2.0.html>. OASIS Committee Specification 03, Aug. 2022.
- [Ros+20] Thomas Rosenstatter et al. “REMIND: A Framework for the Resilient Design of Automotive Systems”. In: *2020 IEEE Secure Development (SecDev)*. 2020, pp. 81–95. doi: 10.1109/SecDev45635.2020.00028.
- [RSK20] Andrew Ramsdale, Stavros Shiaeles, and Nicholas Kolokotronis. “A Comparative Analysis of Cyber-Threat Intelligence Sources, Formats and Languages”. In: *Electronics* 9.5 (2020). doi: 10.3390/electronics9050824.
- [RSL23] Wilhelm Rosinski, Lukas Stahlbock, and Falk Langer. “Container System and Communication Reactivity in a Real-Time Embedded Environment”. In: *2023 IEEE 26th International Symposium on Real-Time Distributed Computing (ISORC)*. 2023, pp. 146–151. doi: 10.1109/ISORC58943.2023.00027.
- [RT19] Thomas Rosenstatter and Katja Tuma. *Deliverable D1.1 A State-of-the-Art Investigation*. Tech. rep. Cyber Resilience in Vehicles (CyRev), Oct. 2019. URL: https://autosec.se/wp-content/uploads/2019/11/CyReV_D1.1_A-State-of-the-Art-Investigation.pdf.
- [Rum+19a] Marcel Rumez et al. “CAN Radar: Sensing Physical Devices in CAN Networks based on Time Domain Reflectometry”. In: *2019 IEEE Vehicular Networking Conference (VNC)*. 2019.

- [Rum+19b] Marcel Rumez et al. “Integration of Attribute-based Access Control into Automotive Architectures”. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. 2019, pp. 1916–1922. doi: 10.1109/IVS.2019.8814265.
- [San+25] Ilya Orson Sandoval et al. *An Attentive Graph Agent for Topology-Adaptive Cyber Defence*. 2025. arXiv: 2501.14700 [cs.LG]. URL: <https://arxiv.org/abs/2501.14700>.
- [Sar+19] Nathan F. Saraiva de Sousa et al. “Network Service Orchestration: A survey”. In: *Computer Communications* 142-143 (2019), pp. 69–94. ISSN: 0140-3664. doi: <https://doi.org/10.1016/j.comcom.2019.04.008>. URL: <https://www.sciencedirect.com/science/article/pii/S0140366418309502>.
- [Sax08] Eric Sax. *Automatisiertes Testen Eingebetteter Systeme in der Automobilindustrie*. Carl Hanser Verlag GmbH & Co. KG, 2008. doi: 10.3139/9783446419018.fm.
- [SB24] Ramakrishnan Subramanian and Ulrich B ker. “Study of Contactless Computer Vision-Based Road Condition Estimation Methods Within the Framework of an Operational Design Domain Monitoring System”. In: *Eng 5.4* (2024), pp. 2778–2804. ISSN: 2673-4117. doi: 10.3390/eng5040145. URL: <https://www.mdpi.com/2673-4117/5/4/145>.
- [SC24] Automotive Information Sharing and Analysis Center. *The Auto-ISAC Launches Automotive Threat Matrix (ATM) Tool to Enhance Vehicle Cybersecurity Governance*. Mar. 2024. URL: <https://automotiveisac.com/press-news/the-auto-isac-launches-automotive-threat-matrix-atm-tool-to-enhance-vehicle-cybersecurity-governance> (visited on 04/03/2024).
- [Sch+20a] Dietmar Scheer et al. “STAR3 - Eine neue Generation der E/E-Architektur”. In: *Sonderprojekte ATZ/MTZ* 25.1 (Dec. 2020),

- pp. 72–79. ISSN: 2509-4610. DOI: 10.1007/s41491-020-0056-5. URL: <https://doi.org/10.1007/s41491-020-0056-5>.
- [Sch+20b] Christoph Schmittner et al. “A Preliminary View on Automotive Cyber Security Management Systems”. In: *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 2020, pp. 1634–1639. DOI: 10.23919/DATE48585.2020.9116406.
- [SDK19] Florian Sommer, Jürgen Dürrwang, and Reiner Kriesten. “Survey and Classification of Automotive Security Attacks”. In: *Information* 10.4 (2019). DOI: 10.3390/info10040148.
- [SEH21] David Schubert, Hendrik Eikerling, and Jörg Holtmann. “Application-Aware Intrusion Detection: A Systematic Literature Review, Implications for Automotive Systems, and Applicability of AutoML”. In: *Frontiers Comput. Sci.* 3 (2021), p. 567873. DOI: 10.3389/FCOMP.2021.567873. URL: <https://doi.org/10.3389/fcomp.2021.567873>.
- [Sha+16] Prateek Sharma et al. “Containers and Virtual Machines at Scale: A Comparative Study”. In: *Proceedings of the 17th International Middleware Conference*. Middleware ’16. Trento, Italy: Association for Computing Machinery, 2016. ISBN: 9781450343008. DOI: 10.1145/2988336.2988337. URL: <https://doi.org/10.1145/2988336.2988337>.
- [Sha24] Kristen Shaw. *How Rivian reduced electrical wiring by 1.6 miles and 44 pounds*. Popular Science. Aug. 6, 2024. URL: <https://www.popsci.com/technology/rivian-zonal-electrical-architecture/> (visited on 01/02/2025).
- [Shi00] Dr. Rob Shirey. *Internet Security Glossary*. Tech. rep. 2828. RFC Editor, May 2000. 212 pp. DOI: 10.17487/RFC2828. URL: <https://www.rfc-editor.org/info/rfc2828>.

- [Sho14] Adam Shostack. *Threat modeling : designing for security*. New York: John Wiley & Sons, Incorporated, 2014. ISBN: 9781306427067.
- [Sik23] Leslie F. Sikos. “Cybersecurity knowledge graphs”. In: *Knowledge and Information Systems* 65.9 (Sept. 2023), pp. 3511–3531. DOI: 10.1007/s10115-023-01860-3.
- [Sil+17] Gary Silberg et al. *Protecting the fleet...and the car business: Today's cyber-physical threats disrupt automotive operating models*. Ed. by KPMG. Dec. 2017. URL: <https://assets.kpmg/content/dam/kpmg/jm/pdf/protecting-the-fleet-webfile.pdf> (visited on 08/18/2023).
- [Sla24] Dirk Slama. *SDV101: Introduction to Software-Defined Vehicles*. 2024. URL: <https://www.sdv.guide/sdv101> (visited on 01/02/2025).
- [SLZ21] Liangliang Sun, Yan-Fu Li, and Enrico Zio. “Comparison of the HAZOP, FMEA, FRAM, and STPA Methods for the Hazard Analysis of Automatic Emergency Brake Systems”. In: *ASCE-ASME J Risk and Uncert in Engrg Sys Part B Mech Engrg* 8.3 (Oct. 2021), p. 031104. ISSN: 2332-9017. DOI: 10.1115/1.4051940. eprint: https://asmedigitalcollection.asme.org/risk/article-pdf/8/3/031104/6765051/risk_008.03.031104.pdf. URL: <https://doi.org/10.1115/1.4051940>.
- [SMC17] Dario Stabili, Mirco Marchetti, and Michele Colajanni. “Detecting attacks to internal vehicle networks through Hamming distance”. In: *2017 AEIT International Annual Conference*. 2017, pp. 1–6.
- [SNI23] Dirk Slama, Achim Nonnenmacher, and Thomas Irawan. *The Software-Defined Vehicle. A Digital-First Approach to Creating Next-Generation Experiences*. O'Reilly, Sept. 2023.
- [SO14] Stefan Seifert and Roman Obermaisser. “Secure automotive gateway - Secure communication for future cars”. In: *2014 12th IEEE*

- International Conference on Industrial Informatics (INDIN)*. 2014, pp. 213–220.
- [Som+24a] Florian Sommer et al. “Combining Cyber Security Intelligence to Refine Automotive Cyber Threats”. In: *ACM Trans. Priv. Secur.* (Feb. 2024). ISSN: 2471-2566. DOI: 10.1145/3644075. URL: <https://doi.org/10.1145/3644075>.
- [Som+24b] Martin Sommer et al. “An Orchestrator for the Dynamic Extension of Automotive E/E Architectures to the Cloud”. In: *Symposium and Summer School on Service-Oriented Computing*. Springer, 2024, pp. 24–41.
- [SS14] Chris Sanders and Jason Smith. *Applied Network Security Monitoring*. Amsterdam: Syngress, an imprint of Elsevier, 2014.
- [ST11] National Institute of Standards and Technology. *Guide for Security-Focused Configuration Management of Information Systems*. Tech. rep. NIST Special Publication (NIST-SP) 800-128. Washington, D.C.: U.S. Department of Commerce, 2011. DOI: 10.6028/NIST.SP.800-128.
- [Sta21] Mirosław Staron. “Automotive Software Development”. In: *Automotive Software Architectures: An Introduction*. Cham: Springer International Publishing, 2021, pp. 1–18. ISBN: 978-3-030-65939-4. DOI: 10.1007/978-3-030-65939-4_1. URL: https://doi.org/10.1007/978-3-030-65939-4_1.
- [Sto21] Hannes Frank Stoll. “Die (re-)konfigurierbare Fahrzeugarchitektur”. German. PhD thesis. Karlsruher Institut für Technologie (KIT), 2021. 271 pp. DOI: 10.5445/IR/1000136041.
- [Str+13] Marius Strobl et al. “Towards automotive virtualization”. In: *2013 International Conference on Applied Electronics*. 2013, pp. 1–6.
- [Str+20] Blake E. Strom et al. *MITRE ATT&CK®: Design and Philosophy*. Tech. rep. McLean, VA: The MITRE Corporation, Mar. 2020. URL: <https://attack.mitre.org/docs/ATTACK.De>

- sign_and_Philosophy_March_2020 . pdf (visited on 01/29/2024).
- [Sub] Subaru. *Subaru Forester 2025 workshop manual*. Accessed via "Free Japan Cars Factory Service Manuals". URL: <https://jdmfsm.info/Auto/Japan/Subaru/Forester/2025/> (visited on 04/01/2026).
- [Suf22] Max-Ferdinand Suffel. "Containerization of AUTOSAR Adaptive - Challenges & Solutions". MICROSAR and DaVinci UserDays. May 2022. URL: https://cdn.vector.com/cms/content/products/microsar/MICROSAR - UserDays/Presentations.2022/Vector.User.Day.2022_MICROSAR.Adaptive.Suffel.Vector.Containerization.of.AUTOSAR.Adaptive - Challenges.and.Solutions.pdf.
- [T-S24] T-Systems. *Automotive security for connected cars*. 2024. URL: <https://www.t-systems.com/de/en/industries/automotive/solutions/automotive-security> (visited on 01/21/2024).
- [Tep+20] Milan Tepić et al. "AutoSec: Multidimensional Timing-Based Anomaly Detection for Automotive Cybersecurity". In: *2020 IEEE 26th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. 2020, pp. 1–10. DOI: 10.1109/RTCSA50079.2020.9203697.
- [The] The MITRE Corporation. *CWE Glossary*. URL: <https://cwe.mitre.org/documents/glossary/>.
- [The24] The Falco Authors, The Linux Foundation. *The Falco Project*. 2024 (version: 0.37.1, initial release 2016). 2024. URL: <https://falco.org/>.
- [Tis18] Mirko Tischer. *The Computing Center in the Vehicle: AUTOSAR Adaptive: Translation of a German publication in Elektronik automotive, special issue "Bordnetz", September 2018*. Ed. by

- Vector Informatik. 2018. URL: https://assets.vector.com/cms/content/know-how/_technical-articles/AUTOSAR/AUTOSAR_Adaptive_ElektronikAutomotive.201809.PressArticle_EN.pdf (visited on 05/10/2020).
- [TMB17] Matthias Traub, Alexander Maier, and Kai L. Barbehön. “Future Automotive Architecture and the Impact of IT Trends”. In: *IEEE Software* 34.3 (2017), pp. 27–32. ISSN: 0740-7459. DOI: 10.1109/MS.2017.69.
- [TSS22] Árpád Török, Zsolt Szalay, and Balázs Sági. “New Aspects of Integrity Levels in Automotive Industry-Cybersecurity of Automated Vehicles”. In: *IEEE Transactions on Intelligent Transportation Systems* 23.1 (2022), pp. 383–391. DOI: 10.1109/TITS.2020.3011523.
- [TW16] Vrizlynn L.L. Thing and Jiaxi Wu. “Autonomous Vehicle Security: A Taxonomy of Attacks and Defences”. In: *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. 2016, pp. 164–170. DOI: 10.1109/iThings-GreenCom-CPSCom-SmartData.2016.52.
- [Ups19] Upstream Security. “Upstream Security Global Automotive Cybersecurity Report 2019”. In: (2019). URL: <https://www.upstream.auto/upstream-security-global-automotive-cybersecurity-report-2019/>.
- [Ups23] Upstream Security. *The Upstream Platform*. 2023. URL: <https://info.upstream.auto/hubfs/OnePager/UpstreamPlatformDataSheet.pdf> (visited on 01/21/2024).
- [Ups24a] Upstream Security. *AutoThreat@Intelligence Cyber Incident Repository*. 2024. URL: <https://upstream.auto/re>

- search/automotive - cybersecurity/ (visited on 01/29/2024).
- [Ups24b] Upstream Security. *Upstream Security Global Automotive Cybersecurity Report 2024*. Ed. by Upstream Security Ltd. 2024. URL: <https://upstream.auto/reports/global-automotive-cybersecurity-report/> (visited on 07/03/2024).
- [Val+24] Sherin Kalli Valappil et al. “Advanced IDPS Architecture for Connected and Autonomous Vehicles”. In: *2024 IEEE Intelligent Vehicles Symposium (IV)*. 2024, pp. 1779–1785. DOI: 10.1109/IV55156.2024.10588659.
- [Van16] Rob Van Os. “SOC-CMM: Designing and Evaluating a Tool for Measurement of Capability Maturity in Security Operations Centers”. MA thesis. Luleå University of Technology, Computer Science, 2016, p. 74. URL: <https://www.soc-cmm.com>.
- [Var19] Marcelino Varas. *AUTOSAR Adaptive in PREEvision*. Webinar. 2019. URL: https://cdn.vector.com/cms/content/events/2019/Webinars19/VectorWebinar_AUTOSAR_Adaptive_PREEvision_20190326.pdf.
- [Vec24] Vector Informatik GmbH. *PREEvision Layer Model*. current version: 10. 2024. URL: <https://www.vector.com/de/de/download/preevision-das-ebenenmodell-hochformat/>.
- [Ven21] BMW i Ventures. Nov. 2021. URL: [Connected%20Vehicle%20Cybersecurity%20Solutions:%20Our%20Investment%20in%20Upstream%20Security](#) (visited on 02/04/2025).
- [Ver+08] Vilhelm Verendel et al. “An Approach to using Honeypots in In-Vehicle Networks”. In: *2008 IEEE 68th Vehicular Technology Conference*. 2008, pp. 1–5. DOI: 10.1109/VETECF.2008.260.

- [Vic23] VicOne. *VicOne Automotive Cyberthreat Landscape Report 2023*. 2023. URL: <https://documents.vicone.com/reports/automotive-cyberthreat-landscape-report-2023.pdf> (visited on 07/03/2024).
- [Vic24] VicOne. *Attack Mapping. Common Threat Techniques Used in Automotive Cyberattacks*. 2024. URL: <https://vicone.com/why-vicone/attack-mapping> (visited on 01/29/2024).
- [Vie+20] Manfred Vielberth et al. “Security Operations Center: A Systematic Study and Open Challenges”. In: *IEEE Access* 8 (2020), pp. 227756–227779. doi: 10.1109/ACCESS.2020.3045514.
- [Vir+24] Apurva Virkud et al. “How does Endpoint Detection use the MITRE ATT&CK Framework?” In: *33rd USENIX Security Symposium (USENIX Security 24)*. 2024, pp. 3891–3908.
- [VS21] Andreas Vetter and Eric Sax. “Hierarchical Versioning to Increase Compatibility in Signal-Oriented Vehicle Networks”. In: *Proceedings of the 27th International Conference on Systems Engineering, ICSEng 2020*. Ed. by Henry Selvaraj, Grzegorz Chmaj, and Dawid Zydek. Cham: Springer International Publishing, 2021, pp. 435–444. ISBN: 978-3-030-65796-3.
- [VVS24] Lars Völker, Josetxo Villanueva, and Jan Schäferling. “Is SOME/IP the Right Solution for the Next 10 Years of Vehicles?” In: *Proceedings of the Automotive Ethernet Congress*. 10th Anniversary. Munich, Germany: Technica Engineering and Ampere, Renault Group, Mar. 2024. URL: https://www.some-ip.com/papers/2024-AEC_Ampere_TechnicaEngineering_-_SOMEIP-next-10-years_final_v1.1.pdf.
- [Was+17] Armin Wasicek et al. “Context-aware intrusion detection in automotive control systems”. In: *Proc. 5th ESCAR USA Conf.* 2017, pp. 21–22.

- [Web15] Marc Weber. *New Communication Paradigms in Automotive Networking: Ethernet and CAN FD are the new trailblazers*. Ed. by Vector Informatik GmbH. Aug. 2015. URL: https://cdn.vector.com/cms/content/know-how/_technical-articles/Ethernet.CANFD.AutomobileElektronik.201508.PressArticle.long.EN.pdf.
- [Web18] Marc Weber. *Automotive Cybersecurity: A Steep Learning Curve*. Vector Congress 2018, Stuttgart, Germany. Nov. 2018. URL: <https://cdn.vector.com/cms/content/events/2018/VeCo18/presentations/VeCo18.Weber.pdf> (visited on 07/23/2024).
- [Web19] Marc Weber. “Untersuchungen zur Anomalieerkennung in automotive Steuergeräten durch verteilte Observer mit Fokus auf die Plausibilisierung von Kommunikationssignalen”. Dissertation. Karlsruhe: KIT, Mar. 2019. doi: 10.5445/IR/1000092815.
- [Wil23] Charles Wilson. *“Autonomous Vehicle Cybersecurity Development Lifecycle”*. Tech. rep. “Version 56”. Motional, Oct. 2023. URL: <https://github.com/nutonomy/AVCDL>.
- [Wu+20] Wufei Wu et al. “A Survey of Intrusion Detection for In-Vehicle Networks”. In: *IEEE Transactions on Intelligent Transportation Systems* 21.3 (2020), pp. 919–933. doi: 10.1109/TITS.2019.2908074.
- [Wu+22] Yujing Wu et al. “Dynamic Rearrangement Compression Algorithm for Intelligent Connected Vehicles”. In: *IEEE Transactions on Vehicular Technology* 71.10 (2022), pp. 10350–10360. doi: 10.1109/TVT.2022.3187469.
- [WWP04] Marko Wolf, André Weimerskirch, and Christof Paar. “Security in automotive bus systems”. In: *Proceedings of Embedded Security in Cars Workshop (ESCAR '04)*. 2004.

- [Yek+23] Ali Recai Yekta et al. “VATT&EK: Formalization of Cyber Attacks on Intelligent Transport Systems - a TTP based approach for Automotive and Rail”. In: *Proceedings of the 7th ACM Computer Science in Cars Symposium*. CSCS '23. Darmstadt, Germany: Association for Computing Machinery, 2023. doi: 10.1145/3631204.3631867. URL: <https://doi.org/10.1145/3631204.3631867>.
- [YT22] Hasan Yasar and Sam E Teplov. “DevSecOps In Embedded Systems: An Empirical Study Of Past Literature”. In: *Proceedings of the 17th International Conference on Availability, Reliability and Security*. ARES '22. Vienna, Austria: Association for Computing Machinery, 2022. ISBN: 9781450396707. doi: 10.1145/3538969.3544451. URL: <https://doi.org/10.1145/3538969.3544451>.
- [Zel+22] Daniel Zelle et al. “ThreatSurf: A method for automated Threat Surface assessment in automotive cybersecurity engineering”. In: *Microprocessors and Microsystems* 90 (2022), p. 104461. ISSN: 0141-9331. doi: <https://doi.org/10.1016/j.micpro.2022.104461>. URL: <https://www.sciencedirect.com/science/article/pii/S0141933122000321>.
- [Zen+21] Jun Zeng et al. “WATSON: Abstracting Behaviors from Audit Logs via Aggregation of Contextual Semantics”. In: *Proceedings of the 28th Annual Network and Distributed System Security Symposium (NDSS)*. Internet Society, Feb. 2021. doi: 10.14722/ndss.2021.24549. URL: https://www.ndss-symposium.org/wp-content/uploads/ndss2021_7A-3-24549_paper.pdf.
- [Zen+22] Jun Zeng et al. “SHADEWATCHER: Recommendation-guided Cyber Threat Analysis using System Audit Records”. In: *2022 IEEE Symposium on Security and Privacy (SP)* (2022), pp. 489–506.

- [Zha+25] Yongheng Zhang et al. “AttacKG+: Boosting attack graph construction with Large Language Models”. In: *Computers & Security* 150 (2025), p. 104220. ISSN: 0167-4048. DOI: <https://doi.org/10.1016/j.cose.2024.104220>. URL: <https://www.sciencedirect.com/science/article/pii/S0167404824005261>.
- [Zwe19] Harald Zweck. *10 Mbps Ethernet Technology and the Challenges Facing Automotive Microcontrollers*. Stuttgart, 2019. URL: https://assets.vector.com/cms/content/events/2019/vAES19/vAES19_01_Zweck_Infineon.pdf (visited on 06/11/2024).

Standards and public guidelines

- [80216a] *IEEE Standard for Ethernet Amendment 1: Physical Layer Specifications and Management Parameters for 100 Mb/s Operation over a Single Balanced Twisted Pair Cable (100BASE-T1)*, 802.3bw-2015. Int. std. IEEE Standards Association, New York, NY, USA, Mar. 7, 2016. DOI: 10.1109/IEEESTD.2016.7433918.
- [80216b] *IEEE Standard for Ethernet Amendment 4: Physical Layer Specifications and Management Parameters for 1 Gb/s Operation over a Single Twisted-Pair Copper Cable*, 802.3bp-2016. Int. std. IEEE Standards Association, New York, NY, USA, Sept. 20, 2016. DOI: 10.1109/IEEESTD.2016.7564011.
- [AUT03] *AUTOSAR - AUTomotive Open System ARchitecture*. AUTOSAR, 2003. URL: <https://www.autosar.org/>.
- [AUT14] *Specification of Secure Onboard Communication*. AUTOSAR Classic Platform, Oct. 2014.
- [AUT16a] *Log and Trace Protocol Specification*. AUTOSAR Foundation, Nov. 2016.

- [AUT16b] *SOME / IP Protocol Specification*. AUTOSAR Foundation, Nov. 2016.
- [AUT17] *AUTOSAR Adaptive Platform*. AUTOSAR, 2017. URL: <https://www.autosar.org/standards/adaptive-platform/>.
- [AUT18] *Specification of Identity and Access Management*. AUTOSAR Adaptive Platform, Mar. 2018.
- [AUT20] *Specification of Intrusion Detection System Protocol*. AUTOSAR Foundation, Nov. 2020.
- [AUT21] *AUTOSAR Feature Model Exchange Format*. AUTOSAR Foundation, Nov. 2021.
- [AUT22] *Requirements on Intrusion Detection System*. AUTOSAR Foundation, Nov. 2022.
- [AUT23a] *Specification of Firewall for Adaptive Platform*. AUTOSAR Adaptive Platform, Nov. 2023.
- [AUT23b] *Specification of Intrusion Detection System Manager for Adaptive Platform*. AUTOSAR Adaptive Platform, Nov. 2023.
- [AUT23c] *Specification of Manifest*. AUTOSAR Adaptive Platform, Nov. 2023.
- [AUT23d] *Specification of Update and Configuration Management*. AUTOSAR Adaptive Platform, Nov. 2023.
- [AUT23e] *Specification of Vehicle Update and Configuration Management*. AUTOSAR Adaptive Platform, Nov. 2023.
- [AUT23f] *Explanation of Firmware Over-The-Air*. AUTOSAR Classic Platform, Nov. 2023.
- [AUT24] *Technical Report on Security Events Specification*. AUTOSAR Foundation, Nov. 2024.

- [IEEC21a] *UN Regulation No. 155 - Cyber security and cyber security management system - Uniform provisions concerning the approval of vehicles with regards to cyber security and cyber security management system*, E/ECE/TRANS/505/Rev.3/Add.154, UN Economic Commission for Europe (UNECE), Mar. 2021. URL: <https://unece.org/sites/default/files/2021-03/R155e.pdf> (visited on 12/10/2024).
- [IEEC21b] *UN Regulation No. 156 - Software update and software update management system - Uniform provisions concerning the approval of vehicles with regards to software update and software updates management system*, E/ECE/TRANS/505/Rev.3/Add.155, UN Economic Commission for Europe (UNECE), Mar. 2021. URL: <https://unece.org/sites/default/files/2024-03/R156e%20%282%29.pdf> (visited on 12/10/2024).
- [Eur18] *Information Sharing and Analysis Center (ISACs) - Cooperative models*. European Union Agency for Cybersecurity, Feb. 2018. URL: <https://www.enisa.europa.eu/publications/information-sharing-and-analysis-center-isacs-cooperative-models>.
- [For23] Forum of Incident Response and Security Teams, 2023. URL: <https://www.first.org/cvss/> (visited on 02/11/2025).
- [IEE12] *IEEE Standard for Configuration Management in Systems and Software Engineering*, IEEE Std 828-2012. Int. std. IEEE Standards Association, 2012.
- [IEE20a] *IEEE Standard for Ethernet - Amendment 5: Physical Layer Specifications and Management Parameters for 10 Mb/s Operation and Associated Power Delivery over a Single Balanced Pair of Conductors*, IEEE 802.3cg-2019. Int. std. IEEE Standards Association, New York, NY, USA, Feb. 5, 2020. doi: 10.1109/IEEESTD.2020.8982251.

- [IEE20b] *IEEE Standard for Ethernet—Amendment 8:Physical Layer Specifications and Management Parameters for 2.5 Gb/s, 5 Gb/s, and 10 Gb/s Automotive Electrical Ethernet*, IEEE 802.3ch-2020. Int. std. IEEE Standards Association, New York, NY, USA, June 30, 2020. doi: 10.1109/IEEESTD.2020.9146430.
- [IEE24] *IEEE Standard for Information Technology—Portable Operating System Interface (POSIX™), The Open Group Standard, Base Specifications, Issue 8*, IEEE Std 1003.1™-2024, IEEE Standards Association, The Open Group, 2024, pp. 1–4107. doi: 10.1109/IEEESTD.2024.10555529.
- [ISO11] *Systems and software engineering — Architecture description*, ISO/IEC/IEEE 42010:2011. Int. std. International Organization for Standardization, International Electrotechnical Commission, IEEE Standards Association, Dec. 2011.
- [ISO15a] *Information technology – IT asset management – Overview and vocabulary*, ISO/IEC 19770-5:2015. Int. std. International Organization for Standardization, International Electrotechnical Commission, 2015.
- [ISO15b] *Road vehicles - Controller area network (CAN) - Part 1: Data link layer and physical signalling*, ISO 11898-1. Int. std. International Organization for Standardization, Dec. 1, 2015.
- [ISO15c] *Software and systems engineering — Reference model for product line engineering and management*, ISO/IEC 26550:2015. Int. std. International Organization for Standardization, International Electrotechnical Commission, Dec. 2015.
- [ISO16a] *Information technology — Reference Architecture for Service Oriented Architecture (SOA RA) — Part 1: Terminology and concepts for SOA*, ISO/IEC 18384-1:2016. Int. std. International Organization for Standardization, International Electrotechnical Commission, June 1, 2016.

- [ISO16b] *ISO/IEC 27035 Information technology — Information security incident management — Part 1: Principles and process*, ISO/IEC 27035-1:2023. Int. std. International Organization for Standardization, International Electrotechnical Commission, Nov. 2016.
- [ISO16c] *Road vehicles - Controller area network (CAN) - Part 2: High-speed medium access unit*, ISO 11898-2. Int. std. International Organization for Standardization, Dec. 1, 2016.
- [ISO16d] *Road vehicles - Local Interconnect Network (LIN) - Part 1: General information and use case definition*, ISO 17987-1. Int. std. International Organization for Standardization, Aug. 1, 2016.
- [ISO17] *Systems and software engineering — Vocabulary*, ISO/IEC/IEEE 24765:2017. Int. std. International Organization for Standardization, International Electrotechnical Commission, Sept. 2017.
- [ISO18a] *Information technology — Security techniques — Information security management systems — Overview and vocabulary*, ISO/IEC 27000:2018. Int. std. International Organization for Standardization, International Electrotechnical Commission, Feb. 1, 2018.
- [ISO18b] *Road vehicles — Functional safety — Part 1: Vocabulary*, ISO 26262-1:2018. Int. std. International Organization for Standardization, Dec. 1, 2018. URL: <https://www.iso.org/standards/std/68383.html> (visited on 12/10/2024).
- [ISO18c] *Road vehicles — Functional safety — Part 3: Concept phase*, ISO 26262-3:2018. Int. std. International Organization for Standardization, Dec. 1, 2018. URL: <https://www.iso.org/standards/std/68385.html>.
- [ISO18d] *Road vehicles — Unified diagnostic services (UDS) — Part 1: Application layer*, ISO 14229-1:2020. Int. std. International Organization for Standardization, Dec. 1, 2018.

- [ISO20a] *Information technology — Information security incident management — Part 3: Guidelines for ICT incident response operations*, ISO/IEC 27035-3:2020. Int. std. International Organization for Standardization, International Electrotechnical Commission, Sept. 2020.
- [ISO20b] *Road vehicles – Safety and cybersecurity for automated driving systems – Design, verification and validation*, ISO/TR 4804:2020. Tech. rep. International Organization for Standardization, Dec. 2020.
- [ISO21] *Road Vehicles - Cybersecurity Engineering*, ISO/SAE 21434:2021. Int. std. International Organization for Standardization, SAE International, Geneva, CH, Aug. 2021.
- [ISO22] *Road vehicles - Safety of the intended functionality*, ISO 21448:2022. Int. std. International Organization for Standardization, Geneva, CH, June 2022. URL: <https://www.iso.org/standard/77490.html>.
- [ISO23a] *Road Vehicles – Test scenarios for automated driving systems – Specification for operational design domain*, ISO 34503:2023. Int. std. International Organization for Standardization, Geneva, CH, Aug. 2023. URL: <https://www.iso.org/standard/78952.html>.
- [ISO23b] *Road vehicles — Software update engineering*, ISO 24089:2023. Int. std. International Organization for Standardization, Geneva, CH, Feb. 2023. URL: <https://www.iso.org/standard/77796.html>.
- [ISO24a] *Information technology – Database languages – GQL*, ISO/IEC 39075:2024. Int. std. International Organization for Standardization, International Electrotechnical Commission, Geneva, CH, Apr. 2024. URL: <https://www.iso.org/standard/76120.html>.

- [ISO24b] *Road Vehicles – Test scenarios for automated driving systems – Scenario categorization*, ISO 34504:2024. Int. std. International Organization for Standardization, Geneva, CH, Feb. 2024. URL: <https://www.iso.org/standard/78953.html>.
- [NIS06] *Guide to computer security log management*, NIST Special Publication (NIST-SP) 800-92. Karen Kent and Murugiah P Souppaya, National Institute of Standards and Technology, U.S. Department of Commerce, 2006.
- [NIS07] *Guide to Intrusion Detection and Prevention Systems (IDPS)*, NIST Special Publication (NIST-SP) 800-94. Karen Scarfone and Peter Mell, National Institute of Standards and Technology, U.S. Department of Commerce, Feb. 2007.
- [NIS11] *NIST Cloud Computing Reference Architecture*, NIST Special Publication 500-292. Fang Liu et al., National Institute of Standards and Technology, U.S. Department of Commerce, Sept. 2011. DOI: <https://doi.org/10.6028/NIST.SP.500-292>.
- [NIS12] *Computer Security Incident Handling Guide*, NIST Special Publication 800-61 Revision 2. Paul Cichonski et al., National Institute of Standards and Technology, U.S. Department of Commerce, Aug. 2012. DOI: <https://doi.org/10.6028/NIST.SP.800-61r2>.
- [NIS16] *Guide to Cyber Threat Information Sharing*, NIST Special Publication (NIST-SP) 800-150. Chris Johnson et al., National Institute of Standards and Technology, U.S. Department of Commerce, Washington, D.C., 2016. DOI: [10.6028/NIST.SP.800-150](https://doi.org/10.6028/NIST.SP.800-150).
- [NIS19] *Glossary of Key Information Security Terms*, NIST Internal Report (NISTIR) 7298 Revision 3. Celia Paulsen and Robert Byers, National Institute of Standards and Technology, U.S. Department of Commerce, Washington, D.C., 2019. URL: <https://csrc.nist.gov/glossary>. DOI: [10.6028/NIST.IR.7298r3](https://doi.org/10.6028/NIST.IR.7298r3).

- [NIS22] *Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities*, NIST Special Publication (NIST-SP) 800-218. Murugiah Souppaya, Karen Scarfone, and Donna Dodson, National Institute of Standards and Technology, U.S. Department of Commerce, Feb. 2022.
- [Obj04] *Data Distribution Service (DDS)*. Object Management Group, Dec. 2004.
- [SAE16] *Cybersecurity Guidebook for Cyber-Physical Vehicle Systems*, SAE J3061. Society of Automotive Engineers, SAE International, Jan. 2016. doi: 10.4271/J3061_201601.
- [SAE18] *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*, SAE J3016, SAE International, June 2018. URL: https://saemobilus.sae.org/content/j3016_201806 (visited on 08/14/2020).
- [Wes+03] *Handbook for Computer Security Incident Response Teams (CSIRTs)*. Moira West-Brown et al., 2003. doi: 10.1184/R1/6574055.v1.