Cyber Resilience for Vehicles

Report type	**Deliverable D2.2**
Report name	**Principles for Resilient Vehicles and smart repairs**
Dissemination level	**Public**
Status	**Final**
Version number	**1.0**
Date of preparation	**2023-12-19**

Authors and Contributors

Editor	**E-mail**
Tomas Olovsson	tomas.olovsson@chalmers.se

Contributors	**E-mail**
Thomas Rosenstatter	tomas.rosenstatter@chalmers.se
Kim Strandberg	kim.strandberg@volvocars.com

The CyReV Consortium

Assured

Chalmers

Combitech

RISE

Volvo Car Corporation

Volvo Technology

Revision chart and history log

Version	Date	Reason
0.1	2021-12-15	First draft of report
0.3	2022-01-24	Draft phase 1
0.5	2023-12-05	First draft phase 2
1.0	2023-12-19	Public version

Table of contents

List of Abbreviations

C-ITS	Cooperative Intelligent Transport Systems
CAN	Controller Area Network
CAN-FD	CAN with Flexible Data-Rate
ECU	Electronic Control Unit
HARA	Hazard analysis and Risk Assessment
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
OEM	Original Equipment Manufacturer
SIEM	Security Information and Event Management
SOC	Security Operations Center
TARA	Threat Analysis and Risk Assessment (security)
V2V	Vehicle to vehicle communication
V2X	Vehicle to everything communication

1 Introduction

This work package report is part of the deliverable "**D2 Development of resilient automotive systems**" and contains a detailed analysis of different resilience techniques and tools available when designing for resilience. Some of these techniques are traditional safety mechanisms which have been used for many years, such as the use of redundant components whereas others are more directed towards cyber-attacks and how to detect and react to malicious activities in vehicles. Some techniques focus on anomaly detection, others on how to always maintain a safe state and guarantee passenger safety, and some focus on long-term resilience in vehicles.

The work in this work package has led to a Ph.D. thesis by Thomas Rosenstatter with the title "On the Secure and Resilient Design of Connected Vehicles: Methods and Guidelines" [3] and to articles published at conferences and in journals: [5][7][8][9][10]. Input from other work packages has also contributed to the Ph.D. thesis, although the main work has taken place here.

It has also led to a Licentiate thesis by Kim Strandberg 2022 "Towards a Secure and Resilient Vehicle Design: Methodologies, Principles and Guidelines" [5], and a Ph.D. thesis to be defended by him early 2024. It has also led to articles published in conferences and journals: [7][9][26][32][33][34].

CyRev has also partly funded Aljoscha Lautenbach towards his Ph.D. degree to be defended early 2024 where he has contributed with two papers: [27][31]. It has also allowed us to collaborate with Karlsruhe Institut für Technologie (KIT) in this project.

In this deliverable, we provide a summary of the contents of the Ph.D. theses and the published papers but do not include the full texts not to infringe copyright rights. To get a complete understanding of the work and to get more details than outlined here, we refer to these publications which are officially available.

The goals for this work package have been to investigate principles for building a resilient vehicle and to identify principles suitable for *detection, mitigation, recovery,* and how to create *endurance* over time. As a result of this work, we have gained knowledge about how to react when potential security events are detected and what mechanisms are available to dynamically reconfigure a vehicle to always offer the best possible service while guaranteeing the safety of its passengers. We have also performed a systematic literature review and identified threats to resilience, categorized them, and mapped them to their corresponding principles and protection mechanisms.

It is important to note that most techniques identified in this work are not limited to cybersecurity. In many cases, it does not matter whether a deviation from normal behavior of a component, subsystem, or system is due to a security problem or if the source is a software or hardware problem. The reactions can, in many cases although not all, be the same. There may be different levels of response, ranging from raising an alarm to be checked by the OEM to immediately enforcing stricter firewall and gateway rules, limiting, or disabling some functionality, disconnecting the vehicle from external communications, or even initiating a complete, safe, shutdown of the vehicle.

In the remainder of this WP, we will use the term *(cyber)**security*** when we refer to intentionally created problems and ***safety*** when we refer to randomly occurring software and hardware problems similar to the traditional work as defined in ISO 26262 [1], even if security problems can endanger the safety of the vehicle.

As described in work package WP1, a vehicle is a complex system with hundreds of computers and multiple networks enabling various types of communication. We are currently transitioning to massively parallel processor systems with thousands of processor cores offering virtualization and AI functionality. There are also multiple ways for vehicles to communicate with the infrastructure around them, for instance USB, Bluetooth and WLAN enable communication to passengers. We have V2X communication to nearby vehicles and road-side objects, and cellular communication to the Internet and cloud-based services. In addition, cameras, radar and other sensors and devices that receive information from the surroundings which also may be faulty, manipulated or made to mis-interpret their input. This makes a vehicle a very complicated system to protect. A systematic approach to both identifying possible threats, failure modes, and how to react to threats is necessary but far from trivial to perform.

1.1 Scope and Purpose

Automated driving functions make decisions based on input from sensors and external communication. Many services are based on machine learning techniques since traditional programming becomes too complicated, and it is impossible to foresee all possible types of situations in advance and how to react to all possible types of input. From a resilience perspective and when facing intelligent attack scenarios, it can be hard to identify the real source of a detected problem and consequently know what principles we should rely on for dynamic reconfiguration of vehicle functions.

The first step in this direction is to define *safe modes* or *safe states* for operating a vehicle. This work has traditionally been part of a safety HARA (hazard and risk analysis) process where the last option is either to enter a “limp home mode” or to completely disable a vehicle. A more fine-grained approach is desirable where functionality is gradually downgraded based on failing sensors, components, or subsystems while always maintaining a safe state of the vehicle. When dealing with cybersecurity issues, it is not obvious that we can reuse the same safe states as those identified in the HARA process. The real source of a problem may be unknown or at least cannot be identified with certainty, thus a state that is identified as safe from a safety perspective may be vulnerable from a security point of view.

In addition, how to react to a problem may differ between safety and security. For example, if a compromised ECU starts transmitting faked speed messages and a monitoring IDS system detects incorrect speed messages on the network, it is not the right action to restart the ECU responsible for transmitting the correct speed messages. Instead, a more complicated process to identify the real source of the problem is needed.

In this work package, we have also looked at IDS systems and how to react when problems are detected, especially since it may be hard for a vehicle to correctly assess its own state and know whether it is functioning well when compromised. We have therefore worked with “reputation systems” where all vehicles report their view of how other vehicles behave and if they seem to function as intended. Such reports can be sent for cloud analysis and if a specific vehicle gets many complaints, a deeper investigation of its functionality is warranted.

1.2 Limitations

More work is needed related to the definition of cyber-secure *safe states*. As an example, we have not performed a complete analysis of how to identify such states but merely identified the need for such work.

Another challenge requiring more work is to identify on what system levels reconfiguration can and should be done: system level, network level or architecture (layers) and how to obtain defense in depth. To do this, it is necessary to identify *failure modes* to get a deeper understanding of how every system and subsystem are related to each other and in which ways they depend on each other's services, and then map failure modes to mitigation techniques.

Much of the work reported in this WP is connected to work done in other work packages. Collaboration between work packages has been a key factor in dealing with such complex problems, and many of the concepts described here are further explored in other work packages as well.

2 Resilient Vehicle Design

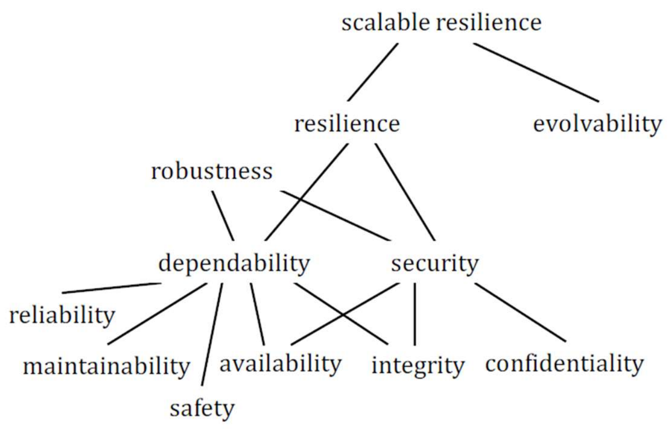
In the CyReV project, we have defined resilience as:

Property of a system with the ability to maintain its *intended* operation in a dependable and secure way, possibly with degraded functionality, in the presence of faults and attacks.

There is also a note to this definition: Dependable and secure refer to attributes such as safety, confidentiality, integrity, privacy and maintainability."

We believe this definition clearly shows what is meant by resilience and gives a clear view of why obtaining resilience is important. In this context, “dependable and secure way” means that we must address safety and security. Security is often referred to as focusing on the three fundamental attributes, i.e., confidentiality, integrity, and availability, whereas dependability is “the ability to avoid service failures that are more frequent and more severe than acceptable” [13]. Therefore, dependability not only includes safety but also other attributes, i.e., availability, reliability, integrity, and maintainability, see the picture below. Laprie [14] further elaborates on the need for ubiquitous systems to maintain dependability despite continuous change, which leads to a similar definition of resilience, namely “the persistence of dependability when facing changes.”

There are also other aspects that are needed for long-term endurance, such as forensics, which is also addressed in this project. The relationship between resilience, security, dependability, and safety can be illustrated as [15]:



The authors differentiate between *resilience* and *scalable resilience,* to highlight that changes, such as technological, functional and environmental, occur over time and thus require the system to be capable of evolving. This emphasis on long-term needs is very important in the automotive domain where vehicles are expected to be operated for several years or even decades.

Due to the relationship between safety, security and resilience, we refer to security when referring to techniques that directly support one of the security attributes. **With resilience, we mean techniques that support both dependability (safety) and security**.

There are many reasons why resilience is important, for instance:

- First and most important, we need to guarantee the safety of passengers and other vehicles due to software, hardware, and security problems. The problems may be security-related but can also result from a hardware or a software problem (“bugs”).
- To make vehicles survive and continue to function, possibly with degraded functionality, when a problem is detected. It may take some time until the problem can be diagnosed by the OEM and before a necessary software update is available. Depending on the result of this analysis, other vehicles may be instructed to reconfigure themselves to prevent this problem from occurring.
- To prevent massive denial of service attacks (DoS) against fleets of vehicles. An attacker who can remotely trigger “limp home mode” or to totally disable vehicles may spawn fleet-wide attacks. Vehicles should be able to reconfigure themselves to survive such attacks without external help.

Many of the services in a vehicle rely on reliable and secure communication and on receiving correct data from sensors and cameras: platooning, virtual traffic lights, and lane changing. In fact, most automated driving functions rely on communication. Therefore, many of the resilience techniques identified in our work focus on communication, networks and the (network) architecture of the vehicle. A current trend in the automotive industry is to move functionality from individual, distributed ECUs to more centralized and powerful multi-processor systems. Still, communication between physical and virtual ECUs are key components in obtaining a safe and secure vehicle.

In this work package, we have investigated how a resilient vehicle system can be built based on the reference architecture from Task 2.1. We have identified components necessary to be able to understand and to enable future simulations of security problems to study possible actions and limit the potential impact. We have also investigated possible ways to react when security problems are detected, and based on functionality, created a usable structure of available methods. In this work package, we have systematically identified the threats to resilience and mapped them to the principles and mechanisms for detection, mitigation, recovery, and endurance.

2.1 Safe states

Resilience is how to guarantee that a vehicle is always in a safe (and secure) state or mode. For example, it may be fully functional, partly functional, or disabled. The identification of safe states is a very complex task and depends largely on the actual design of the vehicle. Therefore, it has not been possible to define specific safe states in this project, nor to answer the question exactly of how many states there should be. Safe states have been defined by ISO, mainly with respect to “traditional safety” without considering security, although the definitions still apply in our broader context:

ISO 26262: “[Safe state is an] ***operating mode (3.102), in case of a failure (3.50), of an item (3.84) without an unreasonable level of risk*** (3.128). Note 1 to entry: See Figure 5. Note 2 to entry: While normal operation can be considered safe, the definition of safe state is only in the case of failure (3.50) in the context of the ISO 26262 series of standards.” [1]

ISO/TR 4804: “[Safe state is an] ***operating mode that is reasonably safe***. Note 1 to entry: The safe state is the state in which both fail-safe (3.18) and fail-degraded (3.16) systems will provide a solution (technically provided by an alternative functionality) to avoid risk, in an acceptable criterion, to any road user (3.46).” [2]

A safe state can be achieved by performing different actions, e.g., to restart an ECU or a subsystem, to reconfigure functionality into a degraded state or to disable one or more services completely. In our REMIND paper (see chapter 3.1), a set of techniques and solutions relevant for the automotive domain are identified, such as reinitialization, reparameterization, relocation/migration, isolation, and software rejuvenation. The main idea is to have a vehicle able to reconfigure itself to achieve graceful degradation when needed.

2.2 How many safe states should we aim for?

The process of defining safe states of a vehicle is complex. It must begin with identifying possible failure modes of the system, which can partly be the result of a Threat Analysis and Risk Assessment process, TARA. It is a complex task, and it is far from obvious whether we should aim for 10 or 100 different states. The number of states is not necessarily the same as clearly identified working modes, but if all deviations from a fully (completely) functional vehicle should be counted as a different state, such as a broken taillight or a non-functioning non-essential ECU, then the number of combinations and the number of states will be huge. It does not mean that we should not care about such deviations, but counting all possible safe states and combinations of deviations is not meaningful since many deviations may not relate to each other, and the impact of deviations can vary vastly.

A more reasonable approach is to focus on different levels when defining safe states. A subsystem may have several safe states which guarantee that its output is trustworthy to its surroundings, but it may also be allowed to produce data of lower quality and notify the surrounding about this. Depending on the quality of data received, some other higher-level functions may have to reduce their functionality or use other sensors or subsystems to double-check or receive more accurate information. Counting the number of possible safe states is therefore complex and may not even be meaningful. However, the functionality of each subsystem must be analyzed in detail in the HARA/TARA processes to make sure the service they deliver is correct and trustworthy, and if it offers limited service, other modules depending on this service should be aware of this and may in turn reduce the service they offer.

For example, if a system receives contradicting input where the reported speed differs between its own sensors and the reported GPS speed, some high-level functions like engaging automatic parking functionality, may have to be disabled until the correct speed can be determined. This discrepancy may also be detected by a network IDS/IPS system which may initiate some internal actions to correct the problem.

There are many issues to consider when safe states are to be identified, for example:

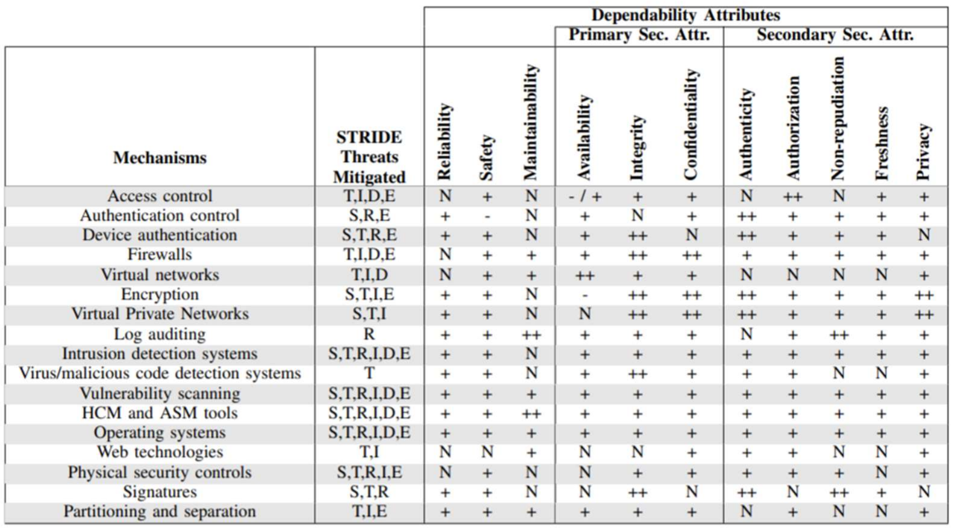
- Do we have similar safe states in safety and security?
- In what way will the transition to more centralized architectures change the scene?
- Safety and security interplay is important: the definition of safe states must be done together regardless of whether we have the same states or not. Security enhancing actions should not be allowed to affect safety in a negative way, and vice versa.
- Standards for safety and security must also be more aligned to allow safety and security design at the same time.

A reference architecture is important when reasoning around these questions where examples of actions and consequences can be analyzed in a limited, simplified environment.

2.3 Security and Safety interplay with respect to safe states

As mentioned above, it is important to investigate the safety and security interplay, since safety and security mechanisms may interfere with each other and cause problems to the other domain, something which is further investigated in work package 3.3. The picture below by Sangchoolie et.al. [20] shows examples of different security mechanisms, threats they address according to the STRIDE methodology [18], and which dependability attribute (safety and security) they may affect (see WP 3.3 for more details). Creating tables like this is useful when analyzing and choosing mechanisms to be used to achieve resilience and whether closer collaboration is needed between engineers in the two fields.

SUMMARY OF THE IMPACT OF COMMON SECURITY MECHANISMS ON DEPENDABILITY AND SECURITY ATTRIBUTES. THE IMPACT IS INDICATED BY ++ (HIGH POSITIVE), + (MEDIUM POSITIVE), N (NEGLIGIBLE) AND - (NEGATIVE) FOR THE CORRESPONDING ATTRIBUTES.



		Dependability Attributes										
					Primary Sec. Attr.			Secondary Sec. Attr.				
Mechanisms	**STRIDE Threats Mitigated**	**Reliability**	**Safety**	**Maintainability**	**Availability**	**Integrity**	**Confidentiality**	**Authenticity**	**Authorization**	**Non-repudiation**	**Freshness**	**Privacy**
Access control	T,I,D,E	N	+	N	- / +	+	+	N	++	N	+	+
Authentication control	S,R,E	+	-	N	+	N	+	++	+	+	+	+
Device authentication	S,T,R,E	+	+	N	+	++	N	++	+	+	+	N
Firewalls	T,I,D,E	N	+	+	+	++	++	+	+	+	+	+
Virtual networks	T,I,D	N	+	+	++	+	+	N	N	N	N	+
Encryption	S,T,I,E	+	+	N	-	++	++	++	+	+	+	++
Virtual Private Networks	S,T,I	+	+	N	N	++	++	++	+	+	+	++
Log auditing	R	+	+	++	+	+	+	N	+	++	+	+
Intrusion detection systems	S,T,R,I,D,E	+	+	N	+	+	+	+	+	+	+	+
Virus/malicious code detection systems	T	+	+	N	+	++	+	+	+	N	N	+
Vulnerability scanning	S,T,R,I,D,E	+	+	+	+	+	+	+	+	+	+	+
HCM and ASM tools	S,T,R,I,D,E	+	+	++	+	+	+	+	+	+	+	+
Operating systems	S,T,R,I,D,E	+	+	+	+	+	+	+	+	+	+	+
Web technologies	T,I	N	N	+	N	N	+	+	+	N	N	+
Physical security controls	S,T,R,I,E	N	+	N	N	+	+	+	+	+	N	+
Signatures	S,T,R	+	+	N	N	++	N	++	N	++	+	N
Partitioning and separation	T,I,E	+	+	+	+	+	+	N	+	N	N	+

An example of non-desired interaction is the use of redundant ECUs where a typical dependability mechanism may compromise security. Consider a vehicle which has two redundant ECUs responsible for reporting the vehicle speed. One ECU is always active and the other passive. If the active ECU stops transmitting messages, the other ECU becomes active and takes over. Then, if the first ECU is restarted and becomes functional again, it remains passive and waits for the other to fail. However, if a third ECU in the vehicle is compromised and starts transmitting incorrect (i.e. faked) speed messages, both legitimate ECUs may believe that the other is active and will refrain from transmitting, leaving only faked messages on the bus. If no redundancy was used at all for speed messages, there would have been conflicting messages on the bus, correct information would be mixed with incorrect information, and receiving nodes would easily detect that something was wrong since speed varies too quickly – for example, every 10ms speed jumps between 50 and 100 km/h.

Similarly, when defining safe states, it may be that some states which are perfectly fine from a safety perspective are less optimal from a security perspective. An attacker may have to take several actions or steps to move the system into such a “safe” state, a state that is less optimal from a security perspective and from which is possible to spawn a certain attack. An example could be to fake speed and some other messages to convince the vehicle it is safe to enter automatic parking when it is actually travelling at high speed on a highway.

2.4 Should the driver be allowed to make some choices?

Another issue related to safe states is if the driver should be allowed to override some alarms, e.g. decide that it is a false alarm, and that the vehicle should not go into “limp home” mode. If we allow driver intervention, it can affect how the vehicle behaves and what safe states it enters. It may be that the vehicle requires assistance to decide whether a function works or not, or it may ask the driver whether some functions should be disabled or not. It may also be possible to allow the vehicle to ask the driver about which sensor value is correct if there is a conflict. This question was raised during the project, but since user actions and behavior are outside the scope of this project, we decided not to investigate it further.

Another related issue is controllability, i.e. how the driver will handle the situation if a vehicle enters a new safe state and needs to immediately shut down some functions. Some mitigation and recovery mechanisms may require an immediate takeover by the driver before they can be executed.

3 A Framework for Resilient Design – Techniques identified

We have investigated the principles for how to respond when suspected security problems are detected. The goal has been to systematically identify the threats to vehicles and map them to principles and mechanisms. This is useful and can guide designers to make an informed and optimal selection of resilience techniques to be used in an automotive system.

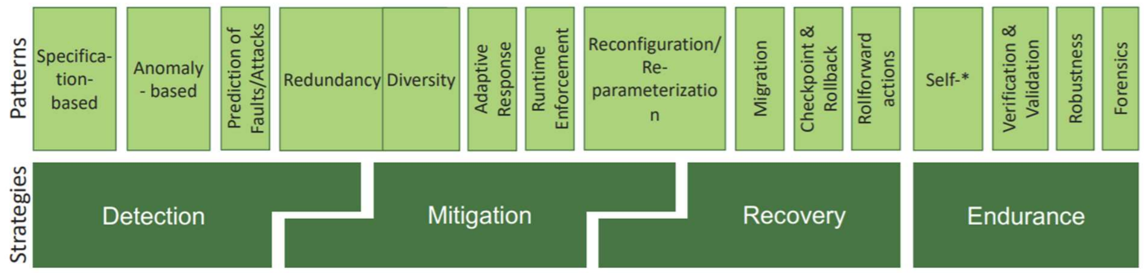
3.1 "REMIND - A Framework for the Resilient Design of Automotive Systems" [7]

Paper presented at IEEE Secure Development (SecDev), Atlanta, GA, USA, 2020

Authors: T. Rosenstatter, K. Strandberg, R. Jolak, R. Scandariato, and T. Olovsson

The complete text is available online and is not duplicated here for copyright reasons.

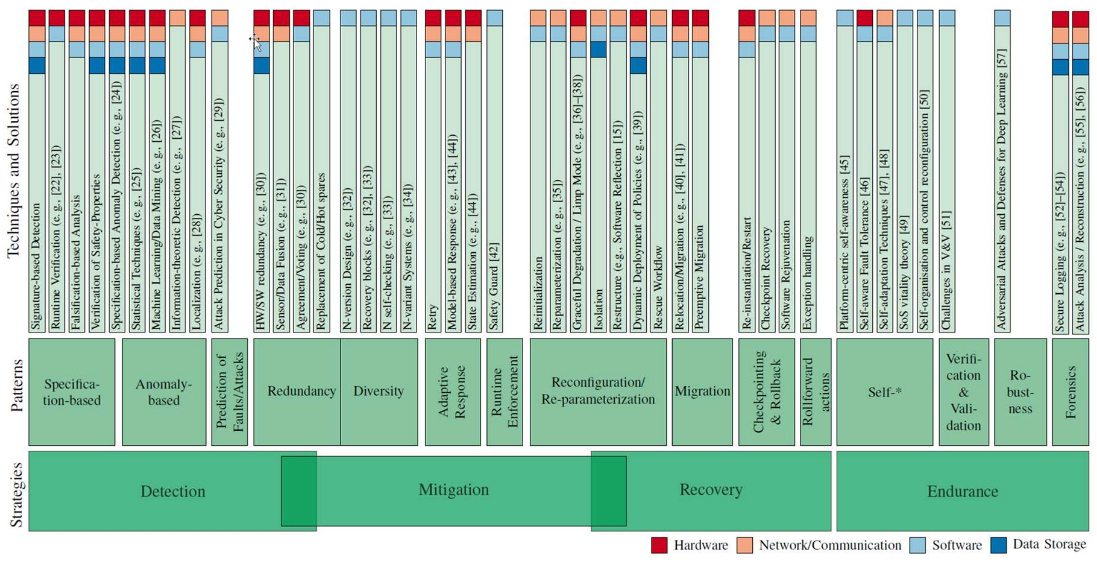
This work has led to the REMIND framework [7] comprising four strategies: *detection*, *mitigation* (analyze and respond), *recovery* and *endurance*. This work required us to perform a structured analysis of resilience techniques found in the literature, where 200 of the most relevant publications indexed by Scopus were selected. As shown in the figure below, we have also created a structure with different "patterns", i.e. more detailed descriptions for how the strategies can be realized:



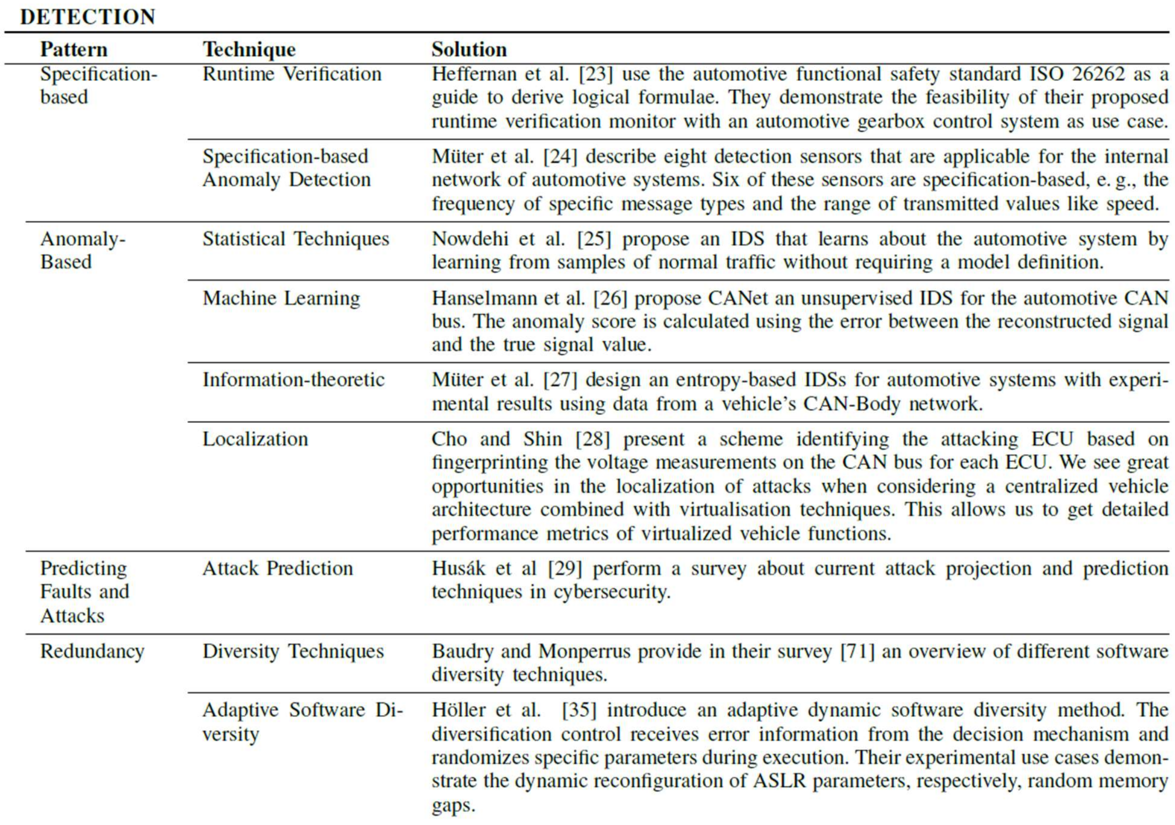
- Some patterns are **twofold**
  - *Redundancy* may *detect & mitigate* anomalies
  - *Reconfiguration* may *mitigate* threats & *recover* the system

We can see that some patterns overlap two strategies, for example redundancy which can be a tool both to detect deviating output by using multiple systems, ECUs, or sensors, but also to mitigate threats by immediately restarting failing components.

The patterns were then further divided into different techniques that can be used to implement a certain strategy, as shown in the figure below. It also shows what type of asset is addressed, *hardware*, *software*, *network communication* or *data storage*. For each technique, one or several important papers are referenced to allow the reader to further investigate the concept:


Techniques and Solutions
Signature-based Detection
Runtime Verification (e. g., [22], [23])
Falsification-based Analysis
Verification of Safety-Properties
Specification-based Anomaly Detection (e. g., [24])
Statistical Techniques (e. g., [25])
Machine Learning/Data Mining (e. g., [26])
Information-theoretic Detection (e. g., [27])
Localization (e. g., [28])
Attack Prediction in Cyber Security (e. g., [29])
HW/SW redundancy (e. g., [30])
Sensor/Data Fusion (e. g., [31])
Agreement/Voting (e. g., [30])
Replacement of Cold/Hot spares
N-version Design (e. g., [32])
Recovery blocks (e. g., [32], [33])
N self-checking (e. g., [33])
N-variant Systems (e. g., [34])
Retry
Model-based Response (e. g., [43], [44])
State Estimation (e. g., [44])
Safety Guard [42]
Reinitialization
Reparameterization (e. g., [35])
Graceful Degradation / Limp Mode (e. g., [36]–[38])
Isolation
Restructure (e. g., Software Reflection [15])
Dynamic Deployment of Policies (e. g., [39])
Rescue Workflow
Relocation/Migration (e. g., [40], [41])
Preemptive Migration
Re-instantiation/Restart
Checkpoint Recovery
Software Rejuvenation
Exception handling
Platform-centric self-awareness [45]
Self-aware Fault Tolerance [46]
Self-adaptation Techniques [47], [48]
SoS vitality theory [49]
Self-organisation and control reconfiguration [50]
Challenges in V&V [51]
Adversarial Attacks and Defenses for Deep Learning [57]
Secure Logging (e. g., [52]–[54])
Attack Analysis / Reconstruction (e. g., [55], [56])
Patterns
Specification-based
Anomaly-based
Prediction of Faults/Attacks
Redundancy
Diversity
Adaptive Response
Runtime Enforcement
Reconfiguration/ Re-parameterization
Migration
Checkpointing & Rollback
Rollforward actions
Self-*
Verification & Validation
Robustness
Forensics
Strategies
Detection
Mitigation
Recovery
Endurance
Hardware
Network/Communication
Software
Data Storage

In the figure, we can see that the *Mitigation* strategy overlaps with *Recovery* and *Detection* and that some techniques and solutions are applicable to both. Appendix B of the REMIND paper contains more details and explanations, for example, it discusses Detection strategies in some detail. A short extract from the paper is shown here:



DETECTION

Pattern	Technique	Solution
Specification-based	Runtime Verification	Heffernan et al. [23] use the automotive functional safety standard ISO 26262 as a guide to derive logical formulae. They demonstrate the feasibility of their proposed runtime verification monitor with an automotive gearbox control system as use case.
	Specification-based Anomaly Detection	Müter et al. [24] describe eight detection sensors that are applicable for the internal network of automotive systems. Six of these sensors are specification-based, e. g., the frequency of specific message types and the range of transmitted values like speed.
Anomaly-Based	Statistical Techniques	Nowdehi et al. [25] propose an IDS that learns about the automotive system by learning from samples of normal traffic without requiring a model definition.
	Machine Learning	Hanselmann et al. [26] propose CANet an unsupervised IDS for the automotive CAN bus. The anomaly score is calculated using the error between the reconstructed signal and the true signal value.
	Information-theoretic	Müter et al. [27] design an entropy-based IDSs for automotive systems with experimental results using data from a vehicle's CAN-Body network.
	Localization	Cho and Shin [28] present a scheme identifying the attacking ECU based on fingerprinting the voltage measurements on the CAN bus for each ECU. We see great opportunities in the localization of attacks when considering a centralized vehicle architecture combined with virtualisation techniques. This allows us to get detailed performance metrics of virtualized vehicle functions.
Predicting Faults and Attacks	Attack Prediction	Husák et al [29] perform a survey about current attack projection and prediction techniques in cybersecurity.
Redundancy	Diversity Techniques	Baudry and Monperrus provide in their survey [71] an overview of different software diversity techniques.
	Adaptive Software Diversity	Höller et al. [35] introduce an adaptive dynamic software diversity method. The diversification control receives error information from the decision mechanism and randomizes specific parameters during execution. Their experimental use cases demonstrate the dynamic reconfiguration of ASLR parameters, respectively, random memory gaps.

Resilience of automotive systems is required to cope with diverse and newly emerging attacks that make use of the advances in communication and functionality. Automotive systems need to maintain the intended functionality, even if degraded, to ensure the safety of the passengers and the surrounding environment. Research in identifying and categorizing resilience techniques has been performed in areas such as cloud computing [16], fog computing [17] and cyber-physical systems[15]. In this work, we review and analyze scientific literature on resilience techniques, fault tolerance, and dependability. As a result, we present the REMIND resilience framework supporting the design of resilient vehicles by (i) identifying techniques for attack detection, mitigation, recovery and resilience endurance; (ii) organizing these techniques into a taxonomy to guide designers in choosing the needed technique for the task at hand; (iii) providing guidelines describing how the proposed taxonomy can be applied against common security threats; and (iv) discussing the trade-os when implementing techniques identified in REMIND.

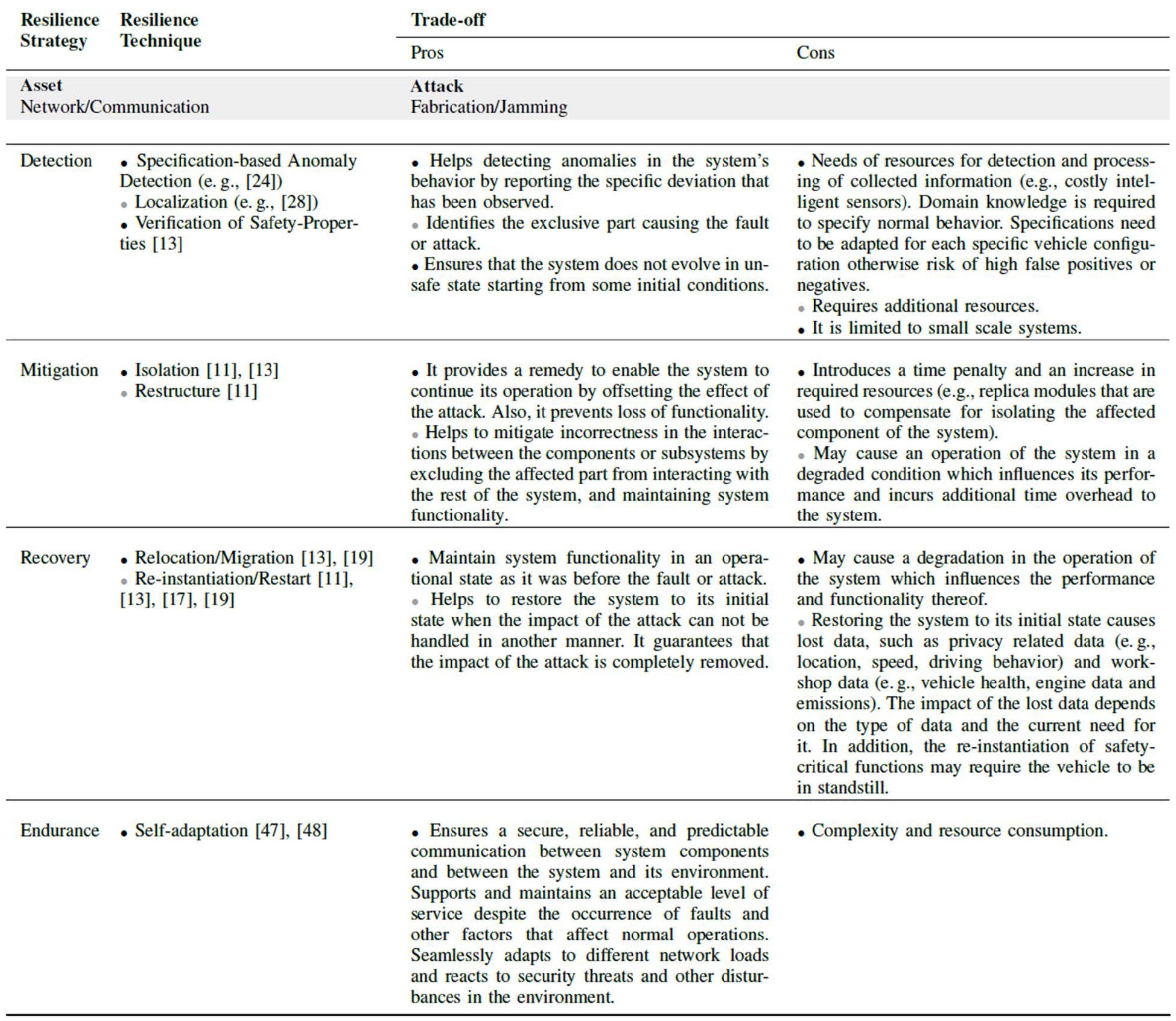
EXTRACT FROM PUBLICATION

Abstract — In the past years, great effort has been spent on enhancing the security and safety of vehicular systems. Current advances in information and communication technology have increased the complexity of these systems and lead to extended functionalities towards self-driving and more connectivity. Unfortunately, these advances open the door for diverse and newly emerging attacks that hamper the security and, thus, the safety of vehicular systems. In this paper, we contribute to supporting the design of resilient automotive systems. We review and analyze scientific literature on resilience techniques, fault tolerance, and dependability. As a result, we present the REMIND resilience framework providing techniques for attack detection, mitigation, recovery, and resilience endurance. Moreover, we provide guidelines on how the REMIND framework can be used against common security threats and attacks and further discuss the trade-offs when applying these guidelines.

CONCLUSION – The reviewed work shows the current research efforts towards making systems resilient to attacks and faults in related domains. We present a novel structure for categorizing resilience techniques in the form of the REMIND framework with the aim to lead designers in making informed decisions when choosing resilience techniques. We build upon the existing work and set the focus on the limitations of automotive systems and their challenges. The REMIND techniques have been chosen considering automotive assets and related attacks which are described in Section III and further linked to the guidelines and trade-off analysis in Appendix A.

Future work includes the validation of the REMIND framework in regard to studying its applicability in industry in more depth. Furthermore, specific solutions for the identified techniques that consider the unique properties of automotive vehicles can be explored. Especially, the role of software-defined networking and its contribution to resilience can be investigated.

Appendix A of this paper contains a detailed 5-page long listing of resilience techniques including trade-offs when implementing a specific technique. An example from the Appendix is shown here which describes fabrication/jamming of network traffic:



Resilience Strategy	Resilience Technique	Trade-off	
		Pros	Cons
Asset Network/Communication		**Attack** Fabrication/Jamming	
Detection	• Specification-based Anomaly Detection (e. g., [24]) • Localization (e. g., [28]) • Verification of Safety-Properties [13]	• Helps detecting anomalies in the system's behavior by reporting the specific deviation that has been observed. • Identifies the exclusive part causing the fault or attack. • Ensures that the system does not evolve in unsafe state starting from some initial conditions.	• Needs of resources for detection and processing of collected information (e.g., costly intelligent sensors). Domain knowledge is required to specify normal behavior. Specifications need to be adapted for each specific vehicle configuration otherwise risk of high false positives or negatives. • Requires additional resources. • It is limited to small scale systems.
Mitigation	• Isolation [11], [13] • Restructure [11]	• It provides a remedy to enable the system to continue its operation by offsetting the effect of the attack. Also, it prevents loss of functionality. • Helps to mitigate incorrectness in the interactions between the components or subsystems by excluding the affected part from interacting with the rest of the system, and maintaining system functionality.	• Introduces a time penalty and an increase in required resources (e.g., replica modules that are used to compensate for isolating the affected component of the system). • May cause an operation of the system in a degraded condition which influences its performance and incurs additional time overhead to the system.
Recovery	• Relocation/Migration [13], [19] • Re-instantiation/Restart [11], [13], [17], [19]	• Maintain system functionality in an operational state as it was before the fault or attack. • Helps to restore the system to its initial state when the impact of the attack can not be handled in another manner. It guarantees that the impact of the attack is completely removed.	• May cause a degradation in the operation of the system which influences the performance and functionality thereof. • Restoring the system to its initial state causes lost data, such as privacy related data (e. g., location, speed, driving behavior) and workshop data (e. g., vehicle health, engine data and emissions). The impact of the lost data depends on the type of data and the current need for it. In addition, the re-instantiation of safety-critical functions may require the vehicle to be in standstill.
Endurance	• Self-adaptation [47], [48]	• Ensures a secure, reliable, and predictable communication between system components and between the system and its environment. Supports and maintains an acceptable level of service despite the occurrence of faults and other factors that affect normal operations. Seamlessly adapts to different network loads and reacts to security threats and other disturbances in the environment.	• Complexity and resource consumption.

Similar tables exist for other types of attacks against the identified asses. For more information, please see the published paper.

4 A framework with resilience techniques

To create a structure of resilience techniques found in the literature and see to what extent they could have prevented known attacks, we have systematically reviewed disclosed attacks targeting vehicles published between 2010 and 2020 and identified what assets were targeted, what security properties were violated and from that we have identified appropriate security and resilience mechanisms that can be useful to mitigate these attacks. Out of a total of 52 unique attacks, 37 high and critical risk attacks were identified.

Focusing on the type of asset being the target for the attack, communication has been attacked most often (29) followed by software (17), sensors (10) and data storage (2). When looking further into attacks towards the communication, we can see that most of the attacks were targeting externally available interfaces (23) if considering interfaces inside the vehicle such as OBD-II as external. By considering only wireless communication, e.g., Bluetooth, key fobs, Wi-Fi and cellular communication, the number of distinct attacks are reduced to 12.

The software asset was most often attacked when it was running (10) followed by attacks when at rest (7) which exploited the software update functionality, weak crypto systems being used or the software state. Attacks towards sensors respectively the hardware, center around showing insecurities in GNSS. However, interesting attacks (5) that target other vehicle sensors, such as the camera, lidar and ultrasonic, have also been published. Most of the attacks on sensors require cyber resilience to withstand attacks such as camera blinding and sensor spoofing. Attacking the data storage seemed to be far less attractive for attackers. However, two attacks have extracted certificates or stolen personal information from replaced spare parts.

We can also see that in 2016 most attacks (15) were published and 7 to 9 attacks per year in the years following. The assets which have been targeted do not show any trends, yet novel attacks on the CAN bus are ceasing.

4.1 "Resilient Shield: Reinforcing the Resilience of Vehicles Against Security Threats" [9]

Paper presented at IEEE 93th Vehicular Technology Conference (VTC2021-Spring), Helsinki, Finland

Authors: K. Strandberg, T. Rosenstatter, R. Jolak, N. Nowdehi, and T. Olovsson

The complete text is available online and is not duplicated here for copyright reasons.

The next step was to develop a threat model by identifying vital vehicle assets that were targeted by these attacks where assets, potential threat actors, and the STRIDE categories for each attack were listed. They were then further mapped to appropriate security and resilience techniques. The security and resilience techniques found in the REMIND framework (chapter 3.1 above) was used and was extended with 7 more categories found useful when categorizing these attacks:

TABLE I
RESILIENT SHIELD. A MAPPING FROM AUTOMOTIVE ASSETS TO IDENTIFIED ATTACKS, POTENTIAL THREAT ACTORS, STRIDE THREAT CATEGORIES AND ULTIMATELY TO APPROPRIATE SECURITY AND RESILIENCE TECHNIQUES, AND SECURITY GOALS (SGS).

Isolation
Availability
Authorization
Confidentiality
Maintainability
Authenticity
Freshness
Privacy
Integrity
Reliability

🟩 Resilience patterns identified in REMIND [18]

Assets targeted by attacks with high or critical risk. *ToE category:subcategory reference*	**Potential Threat Actors** *Financial Actor (FA) Foreign Country (FC) Cyber Terrorist (CT) Insider (IN) Hacktivist (HA) Script Kiddie (SK)*	**STRIDE categories** *(S)poofing (T)ampering (R)epudiation (I)nformation Disclosure (D)enial of service (E)levation of privilege*	(SG.1,8) Authentication	(SG.1) Encryption	🟩 (SG.2) Redundancy/Diversity	(SG.3) Access Control	🟩 (SG.3) Runtime Enforcement	(SG.4,8) Secure Storage	(SG.4) Secure Boot	(SG.4) Secure Programming	(SG.4) Secure Software Update	🟩 (SG.4) Verification & Validation	🟩 (SG.5) Separation	🟩 (SG.6) Specification-based Detection	🟩 (SG.6) Anomaly-based Detection	🟩 (SG.6) Prediction of Faults/Attacks	🟩 (SG.6) Adaptive Response	🟩 (SG.6) Reconfiguration	🟩 (SG.6) Migration	🟩 (SG.6) Checkpoint & Rollback	🟩 (SG.6) Rollforward actions	🟩 (SG.7) Self-X	🟩 (SG.7) Robustness	🟩 (SG.8) Forensics
Hardware																								
sensor:camera [34], [35]	FC, CT, HA	S, D			●												●	●				●	●	
sensor:GNSS [24], [26], [29], [30], [32]	FC, CT, HA	S	●		●										●		●	●				●	●	
sensor:lidar [28], [34]	FC, CT, HA	S, D			●												●	●				●	●	
sensor:ultrasonic [35]	FC, CT, HA	S, D			●												●	●				●	●	
Communication																								
internal:can [40], [44], [46], [47], [49]	FA, FC, CT, IN, HA	S, T, I, D	●	●	●	●	●						●	●	●	●		●	●			●		●
internal:flexray [37]	FA, FC, CT, HA	S, D	●			●	●						●	●	●			●				●		●
external:bluetooth [4], [36]	FC, CT, HA	S, T, D, E	●			●							●											
external:usb [4]	FC, CT, HA	S, T, E	●			●							●											
external:keyfob [22], [23]	HA, SK	S	●			●									●								●	
external:wifi [5], [33]	HA, SK	S, I	●	●		●				●			●											
external:cellular [3], [4], [41], [45], [51], [52]	FC, CT, HA, SK	S, T, I, D, E	●			●							●											
external:obdII [7], [27], [31], [38], [40], [43], [46], [48]	CT, HA	S, T, I, D, E	●			●	●						●	●	●			●		●		●		●
external:debugport [3], [41]	HA, IN	I, E	●			●																		
Software																								
running:state [25]	FC, CT, HA	S, D				●				●				●	●					●	●			●
running:firmware [3]–[5], [33], [36], [39], [41], [45], [51], [52]	FC, CT, HA	S, T, E				●				●	●	●	●	●	●			●				●		●
instorage:update [4], [36], [41]	HA, SK	S, T, E	●	●		●		●	●		●		●	●	●	●	●	●	●					●
instorage:weakcrypto [21], [50], [52]	FC, CT, HA, SK	S, E	●							●													●	
Data Storage																								
crypto:certificates [41]	FC, CT, HA	I		●		●		●	●															
hw:replaced [42]	HA, SK	I	●	●		●																		

Existing work on securing vehicles focuses on providing frameworks that help designers and developers identify the necessary mechanisms to mitigate various attack scenarios. Microsoft STRIDE [18], for instance, provides a tool for threat modelling. HEAVENS [19] supports developers in defining security objectives based on their proposed TARA. Other works such as Sommer et al. [21], focus on a taxonomy for attacks against automotive systems. This chapter combines security and resilience techniques needed in automotive systems in one framework. For the proposed framework, we apply the SPMT methodology on systematically identified attacks to derive security guidelines and detailed directives focusing on security and resilience. We further map the potential threat actors to the assets exposed by each attack and show which security and resilience techniques can be deployed to mitigate them. The resulting framework, named Resilient Shield, builds the base for designing secure and resilient systems, yet allows them to be easily extended in the presence of novel attacks.

EXTRACT FROM PUBLICATION

Abstract — Vehicles have become complex computer systems with multiple communication interfaces. In the future, vehicles will have even more connections to e.g., infrastructure, pedestrian smartphones, cloud, road-side units, and the Internet. External and physical interfaces, as well as internal communication buses have shown to have potential to be exploited for attack purposes. As a consequence, there is an increase in regulations which demand compliance with vehicle cyber resilience requirements. However, there is currently no clear guidance on how to comply with these regulations from a technical perspective.

To address this issue, we have performed a comprehensive threat and risk analysis based on published attacks against vehicles from the past 10 years, from which we further derive necessary security and resilience techniques. The work is done using the SPMT methodology where we identify vital vehicle assets, threat actors, their motivations and objectives, and develop a comprehensive threat model. Moreover, we develop a comprehensive attack model by analyzing the identified threats and attacks. These attacks are filtered and categorized based on attack type, probability, and consequence criteria. Additionally, we perform an exhaustive mapping between asset, attack, threat actor, threat category, and required mitigation mechanism for each attack, resulting in a presentation of a secure and resilient vehicle design. Ultimately, we present the Resilient Shield, a novel and imperative framework to justify and ensure security and resilience within the automotive domain.

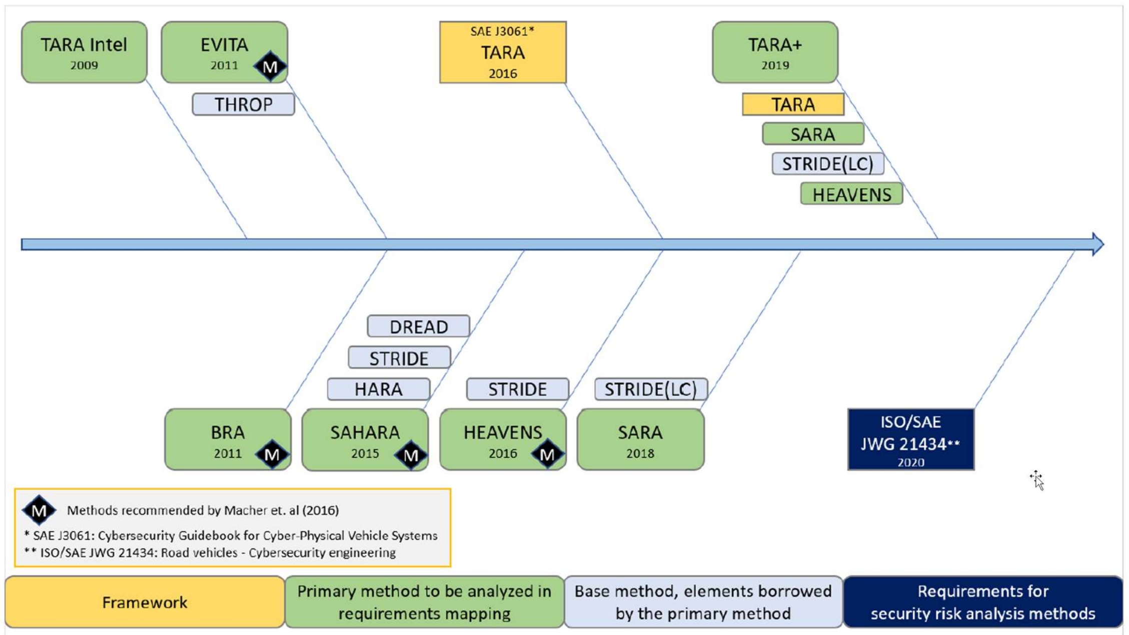
CONCLUSION -- We have performed a comprehensive threat and risk analysis of published attacks against vehicles and derived imperative security and resilience mechanisms by applying the SPMT methodology. A threat model with vital vehicle assets and related potential threat actors, their motivations, and objectives, was developed. By an extensive analysis of threats and attacks, further filtered and categorized based on attack type, probability and consequence criteria, an attack model was developed based on the remaining high-risk attacks. Based on the developed models, a comprehensive mapping between asset, attack, threat actor, threat category, and defense mechanisms was performed for all attacks and is presented in Table I. Table I summarizes the outcomes by applying SPMT, i.e. the Resilient Shield, a novel framework both justifying and defining imperative security and resilient mechanisms needed in a modern vehicle. Consequently, the Resilient Shield can be used as a vital baseline for protection against common security threats and attacks.

We believe our work is imperative for facilitating and guiding the design of resilient automotive systems; however, it still remains to be seen how large the coverage is in relation to future attacks. Moreover, testing and validation of the Resilient Shield within an industrial context is left as a future work.

5 Risk assessment and Standardization

Standardization of security work in the vehicular domain is important, and in an earlier project, HEAVENS, we developed a framework for a future standard with respect to risk assessment with a focus on cybersecurity. The major goal with this project was to develop a threat and risk assessment method which addresses security which is aligned with established functional safety standards for road vehicles (ISO 26262). This led to the HEAVENS security model which is used in the automotive industry today and it has heavily influenced the SAE J3061 and ISO/SAE 21434 standards which, as of 2023, are used by the industry to reach type approval.

The picture below shows projects and work which has influenced the ISO/SAE 21434 standard and shows the role of the HEAVENS model [25]:


TARA Intel 2009
EVITA 2011 M
THROP
SAE J3061* TARA 2016
TARA+ 2019
TARA
SARA
STRIDE(LC)
HEAVENS
DREAD
STRIDE
HARA
STRIDE
STRIDE(LC)
BRA 2011 M
SAHARA 2015 M
HEAVENS 2016 M
SARA 2018
ISO/SAE JWG 21434** 2020
M Methods recommended by Macher et. al (2016)
* SAE J3061: Cybersecurity Guidebook for Cyber-Physical Vehicle Systems
** ISO/SAE JWG 21434: Road vehicles - Cybersecurity engineering
Framework
Primary method to be analyzed in requirements mapping
Base method, elements borrowed by the primary method
Requirements for security risk analysis methods

Since the ISO/SAE 21434 standard introduces additional requirements which were not present in HEAVENS for the risk assessment process, the HEAVENS standard needs to be updated. We have therefore in this project updated the requirements to become HEAVENS 2.0 to facilitate for those who currently use it in practice. HEAVENS 2.0 works as a drop-in threat analysis and risk assessment (TARA) model for ISO/SAE 21434. Practitioners who are already familiar with HEAVENS 1.0 will be able to learn this model easily and therefore be one step closer to applying ISO/SAE 21434. Finally, with minor parameter calibrations, HEAVENS 2.0 can also be applied to similar industries, such as medical devices or industrial systems.

This update aligns terminology, it merges threat analysis and risk assessment phases and includes damage scenario identification. It further adjusts the threat levels to meet the requirements in the ISO standard to require a specific number and names for the attack feasibility rating. Finally, a change is that HEAVENS 2.0 focuses on the road user as the primary stakeholder and no longer on the OEM perspective which affects the impact rating of vulnerabilities and threats. All in all, 12 updates to the HEAVENS 1.0 framework/standard were introduced.

ISO/SAE 21434 has quickly been established in the industry. However, at least two significant challenges remain. In terms of the management of TARA throughout the life cycle and the supply chain, as well as in the response to vulnerabilities and incidents, the standard should be improved and extended. We have therefore performed a methodological gap analysis to identify and discuss challenges and issues, based on which we propose two major improvements to ISO/SAE 21434 that could be incorporated in future versions of the standard: A novel management process for TARA to improve risk management over the life cycle and supply chain; and a revised process for identifying and responding to vulnerabilities and attacks that is better aligned with established processes and more practically feasible.

5.1 “Proposing HEAVENS 2.0 – an automotive risk assessment model” [27]

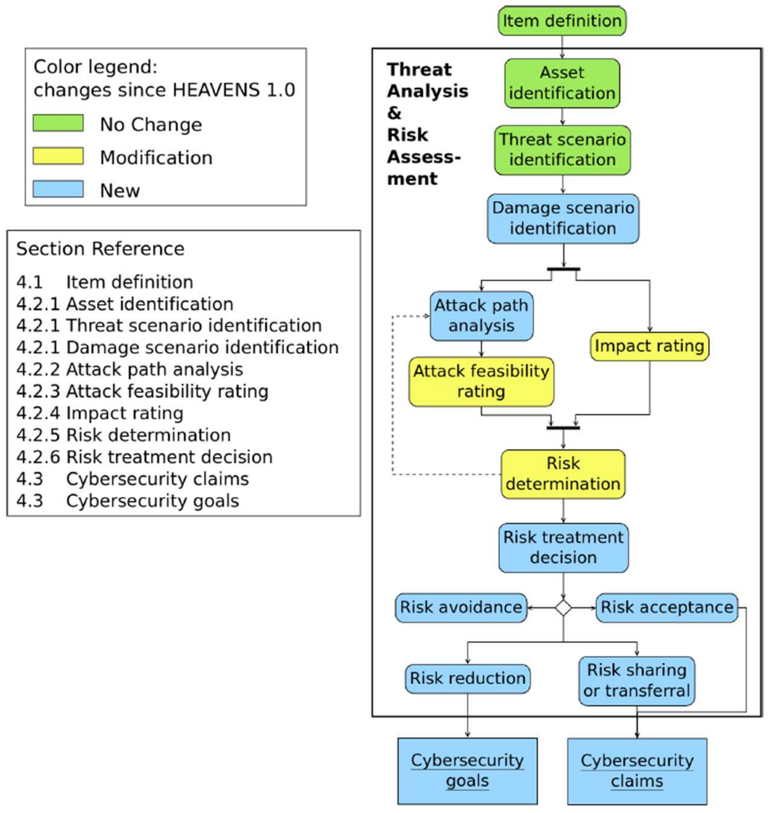
Paper presented at: ACM Computer Science in Cars Symposium, CSCS ‘21

Authors: A. Lautenbach, M. Almgren, T. Olovsson

The complete text is available online and is not duplicated here for copyright reasons.

The update of HEAVENS 1.0 aligns terminology, it merges threat analysis and risk assessment phases and includes damage scenario identification. It further adjusts the threat levels to meet the requirements in the ISO standard to require a specific number and names for the attack feasibility rating. Finally, a change is that HEAVENS 2.0 focuses on the road user as the primary stakeholder and no longer on the OEM perspective which affects the impact rating of vulnerabilities and threats. All in all, 12 updates to the HEAVENS 1.0 framework/standard were introduced.

The workflow of the HEAVENS 2.0 can be summarized as:


Color legend:
changes since HEAVENS 1.0
No Change
Modification
New
Section Reference
4.1 Item definition
4.2.1 Asset identification
4.2.1 Threat scenario identification
4.2.1 Damage scenario identification
4.2.2 Attack path analysis
4.2.3 Attack feasibility rating
4.2.4 Impact rating
4.2.5 Risk determination
4.2.6 Risk treatment decision
4.3 Cybersecurity claims
4.3 Cybersecurity goals
Item definition
Threat Analysis & Risk Assess-ment
Asset identification
Threat scenario identification
Damage scenario identification
Attack path analysis
Impact rating
Attack feasibility rating
Risk determination
Risk treatment decision
Risk avoidance
Risk acceptance
Risk reduction
Risk sharing or transferral
Cybersecurity goals
Cybersecurity claims

EXTRACT FROM PUBLICATION

Abstract – Risk-based security models have seen a steady rise in popularity over the last decades, and several security risk assessment models have been proposed for the automotive industry. The new UN vehicle regulation 155 on cybersecurity provisions for vehicle type approval, as part of the 1958 agreement on vehicle harmonization, mandates the use of risk assessment to mitigate cybersecurity risks and is expected to be adopted into national laws in 54 countries within 1 to 3 years. This new legislation will also apply to autonomous vehicles. The automotive cybersecurity engineering standard ISO/SAE 21434 is seen as a way to fulfill the new UN legislation, so we can expect quick and wide industry adoption. One risk assessment model that has gained some popularity and is in active use in several companies is the HEAVENS model, but since ISO/SAE 21434 introduces additional requirements on the risk assessment process, the original HEAVENS model does not fulfill the standard.

In this paper, we investigate the gap between the HEAVENS risk assessment model and ISO/SAE 21434, and we identify and propose 12 model updates to HEAVENS to close this gap. We also discuss identified weaknesses of the HEAVENS risk assessment model and propose 5 additional model updates to overcome them. In accordance with these 17 identified model updates, we propose HEAVENS 2.0, a new risk assessment model based on HEAVENS which is fully compliant with ISO/SAE 21434

Conclusions – Thanks to new legislation, ISO/SAE 21434 will see widespread adoption in industry and automotive companies need to learn how to integrate cybersecurity processes on project and organizational level. Threat analysis and risk assessment is one of the most prominent of these processes, and it seems especially prudent to apply Proposing HEAVENS 2.0 in autonomous driving use cases to minimize the potential for maliciously caused safety incidents.

In order to facilitate the continued use of experiences from HEAVENS 1.0 in automotive projects, we analyzed its gap to the risk assessment framework mandated by ISO/SAE 21434. Consequently, we proposed 12 model updates to close this gap, and we also addressed 5 shortcomings identified for HEAVENS 1.0. Together, these 17 model updates form the basis for HEAVENS 2.0.

HEAVENS 2.0 works as a drop-in threat analysis and risk assessment (TARA) model for ISO/SAE 21434. Practitioners who are already familiar with HEAVENS 1.0 will be able to learn this model easily and therefore be one step closer to applying ISO/SAE 21434. Finally, with minor parameter calibrations, HEAVENS 2.0 can also be applied to similar industries, such as medical devices or industrial systems.

5.2 Gap analysis of ISO/SAE 21434 – Improving the automotive cybersecurity engineering life cycle” [31]

Paper presented at 2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC 2023)

Authors: D. Grimm, A. Lautenbach, M. Almgren, T. Olovsson

The complete text is available online and is not duplicated here for copyright reasons.

Abstract – Due to the ongoing legislative shift towards mandated cybersecurity for road vehicles, the automotive cybersecurity engineering standard ISO/SAE 21434 is seeing fast adoption throughout the industry. Early efforts focus on threat analysis and risk assessment (TARA) in the concept and development phases, exposing the challenge of managing TARA results coherently throughout the supply chain and life cycle.

While the industry focuses on TARA, other aspects such as vulnerability or incident handling are receiving less attention. However, the increasing threat landscape makes these processes increasingly important, posing another industry challenge.

In order to better address these two challenges, we analyze the cybersecurity engineering framework of ISO/SAE 21434 for gaps or deficiencies regarding TARA management and vulnerability and incident handling, as well as similar processes for incident handling in IT security. The result is a proposal for modifications and augmentations of the ISO/SAE 21434 cybersecurity engineering framework. In particular, we propose a TARA management process to facilitate the coordination and information exchange between different systems and life cycle phases, and we propose improvements to the vulnerability and incident handling processes in ISO/SAE 21434 so that they are more aligned with established standards. This amounts to 13 new terminology definitions, 4 new process steps, 2 modified process steps and 1 entirely new process.

Conclusions – There is a need for clear structures and guidelines around automotive cybersecurity engineering, and ISO/SAE 21434 is largely fulfilling that need. Nevertheless, there are aspects that can and should be improved, in particular around the interaction of TARA processes and other cybersecurity activities, such as vulnerability and incident handling. In line with this, we have proposed a new TARA management process and improvements to the vulnerability and incident handling processes in ISO/SAE 21434, building on existing IT standards and guidelines, as well as on research into TARA improvements. We expect that our proposed improvements will help automotive companies to better coordinate their cybersecurity activities, and that an adoption of the proposed terminology will lead to improved clarity in communication. Hopefully, they can be considered in future versions of ISO/SAE 21434.

6 Attacks and Attack Detection

We have investigated some detection mechanisms for in-vehicle networks and this work is closely related to WP5. Here we have focused on usability with respect to resilience, and particularly the detection mechanisms described earlier. First, a Bachelor Thesis at Chalmers has resulted in a book chapter describing in-vehicle communication and security [10].

We have also recognized a work performed at Chalmers where a new type of IDS system was developed, and we wanted to see to what extent this method would be applicable and could be tweaked to vehicular networks. This work resulted in SPECTRA; a system based on spectral analysis of CAN message payloads. It has been implemented and tested in a Volvo XC 60 vehicle and the results are very promising (see chapter 6.2).

6.1 Published book chapter: "Security of In-Vehicle Communication Systems" [10]

Chapter appearing in Decision Support Systems and Industrial IoT in Smart Grid, Factories, and Cities.

Authors: D. Dubrefjord, M. Jang, H. Hadi, T. Olovsson

The complete text is available online and is not duplicated here for copyright reasons.

Abstract - The automotive industry has seen remarkable growth in the use of network and communication technology. These technologies can be vulnerable to attacks. Several examples of confirmed attacks have been documented in academic studies, and many vehicular communications systems have been designed without security aspects in mind. Furthermore, all the security implications mentioned here would affect the functionality of decision support systems (DSS) of IoT and vehicular networks. This chapter focuses on in-vehicle security and aims to categorize some attacks in this field according to the exploited vulnerability by showing common patterns. The conclusion suggests that an ethernet-based architecture could be a good architecture for future vehicular systems; it enables them to meet future security needs while still allowing network communication with outside systems.

CONCLUSIONS – The analysis of the in-vehicle communication protocols CAN, LIN, FlexRay, and Automotive Ethernet has shown that all protocols are developed with hardly any thought of security in mind. Some countermeasures or fixes have been listed for the different protocols, but the simplicity of the protocols often makes it impractical or impossible to apply the fixes. The four above mentioned protocols all lack security features for authentication, which makes it possible to, for example, perform replay-attacks and spoof messages. This lack of security has broader implications since vehicles are now becoming connected to the Internet. A vulnerability may be remotely exploited allowing an attacker to control arbitrary functions of a vehicle and at will disrupt the rightful functionality. The authors believe that a transition to automotive Ethernet is an important step forward for securing future connected vehicles.

Ethernet is a well-researched area when it comes to security, and it makes it possible to transfer well-known protocols and security technologies to vehicular networks. The cost of Ethernet is what currently makes it unattractive, but the authors believe prices will go down when it becomes a more mature and commonly used technology in our vehicles.

6.2 “Spectra: Detecting Attacks on In-Vehicle Networks through Spectral Analysis of CAN-Message Payloads” [12]

Paper presented at 36th ACM Symposium on Applied Computing, Gwangju, South Korea 2021

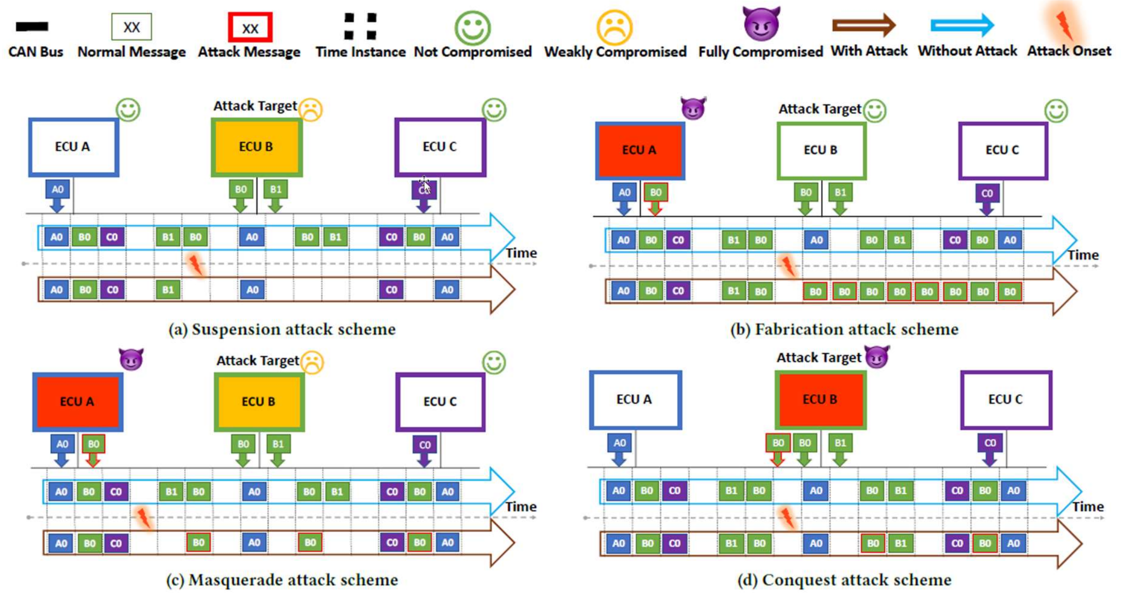
Authors: W. Aoudi, M. Almgren, N. Nowdehi, T. Olovsson

The complete text is available online and is not duplicated here for copyright reasons.

The problem with current in-vehicle IDS systems is their rate of false alarms. Good systems can have a seemingly low false alarm rate, such as 10^{-6}, but in vehicular environments this is decades away from being useful. One false alarm per car per year means millions or tens of millions of events to investigate every year for an OEM.

We have developed SPECTRA, which combines a high detection rate with extremely few false alarms – much better than most other proposed solutions. The SPECTRA system has many promising features although further testing in real environments is needed to fully see its advantages and possible disadvantages.

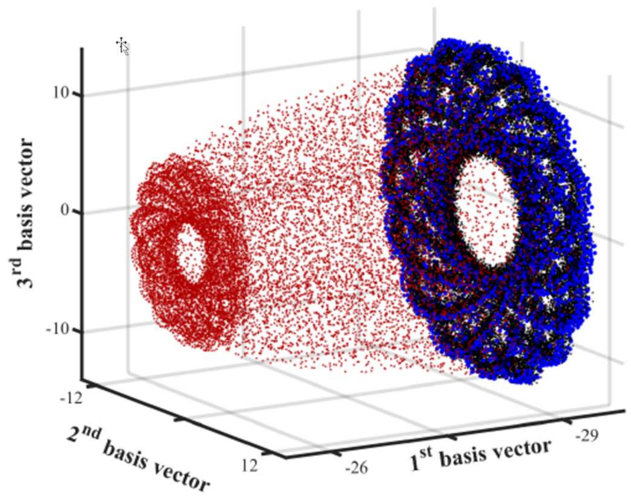
Four different attack scenarios were tested: *suspension attacks, fabrication attacks, masquerade attacks and conquest attacks*:



Schematics for a *suspension* attack (a), *fabrication* attack (b), *masquerade* attack (c), and *conquest* attack (d).

The conquest attack (figure d) is the hardest attack type to deal with by an IDS system. In a conquest attack, the adversary directly conquers the target ECU by fully compromising it. The adversary is able to reprogram the target ECU instead of having to compromise another node on the network to inject the intended malicious payload. Unlike the other scenarios (a-c), this attack causes no changes in the normal behavior of any of the ECUs with respect to message frequency, clock offset, or clock skew behavior. SPECTRA was still able to detect this change of behavior and only needs a short learning period to understand the system it should monitor.

When performing a masquerade attack, the deviation from normal behavior can be visualized as follows:



In short, the blue vector (circle) is the result of the training phase, and all regular network traffic should fall into this cluster of blue vectors. When under attack, the behavior changes and the traffic will look like the red set of vectors. For a full explanation and more details about how the system works, please see the published paper. The characteristics of SPECTRA are promising and is specification-agnostic which makes it applicable to a wide range of vehicle models and makes it deployable in real-world settings, something we tested in a real Volvo XC60 vehicle.

EXTRACT FROM PUBLICATION

Abstract - Nowadays, vehicles have complex in-vehicle networks that have recently been shown to be increasingly vulnerable to cyber-attacks capable of taking control of the vehicles, thereby threatening the safety of the passengers. Several countermeasures have been proposed in the literature in response to the arising threats, however, hurdle requirements imposed by the industry are hindering their adoption in practice. In this paper, we propose spectra, a data-driven anomaly-detection mechanism that is based on spectral analysis of CAN-message payloads. Spectra does not abide by the strict specifications predefined for every vehicle model and addresses key real-world deployability challenges.

CONCLUSION – With the rapid increase in the number of cyberattacks on vehicles, designing intrusion detection systems for CAN communication has become a major area of interest. This paper has made several noteworthy contributions to the field of automotive security. First, we have presented spectra, an efficient attack-detection mechanism that is particularly suitable for the IVN domain. Second, we have demonstrated, through extensive experiments including performing attacks on a 2018 Volvo XC60 test vehicle, how SPECTRA can detect stealthy attacks on IVNs. Finally, we have shown that SPECTRA enjoys the advantage of being specification-agnostic, which makes it applicable to a wide range of vehicle models and deployable in real-world settings.

7 Secure software updates and reference architectures

Having a secure software update process is essential to be able to guarantee a fully functional and resilient system design. Software updates should be possible to perform at any time using any type of network connection. We have created a unified software update framework, UniSUF, which should fulfill most demands for a versatile, flexible, and secure solution. We have also developed requirements for secure vehicle software updates by defining an attacker model and from there derived security requirements. The resulting framework can be used as a reference architecture to guide when engineers and software architects design software update systems, not only for vehicles but also for related areas such as cyber-physical systems, IoT devices and smart cities.

It is important to note that every vehicle is unique and needs to have its unique software packages. Thus, a unique vehicle configuration, multiple software files for every ECU, many unique cryptographic keys, and ECU-specific diagnostic requests are required. For instance, special cryptographic keys are needed to turn off security functionality that might otherwise block the installation process.

There are three main entities involved in the software update process: the producer, the consumer, and the repository. The producer is responsible for producing the software. The consumer is responsible for the download and installation process of the software, and the repository is a storage point for software preferably located in various cloud sources, enabling both proximity and redundancy for data in relation to the vehicle.

7.1 Secure Vehicle Software Updates: Requirements for a Reference Architecture [32]

Paper presented at IEEE Vehicular Technology Conference VTC-2023, 979-8-3503-1114-3 (ISBN).

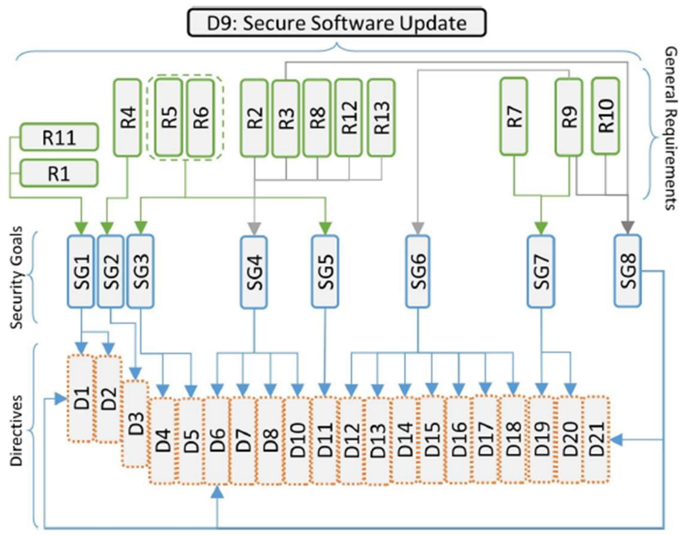
Authors: K. Strandberg, U. Arnljung, T. Olovsson, and D. Kengo Oka

The complete text is available online and is not duplicated here for copyright reasons.

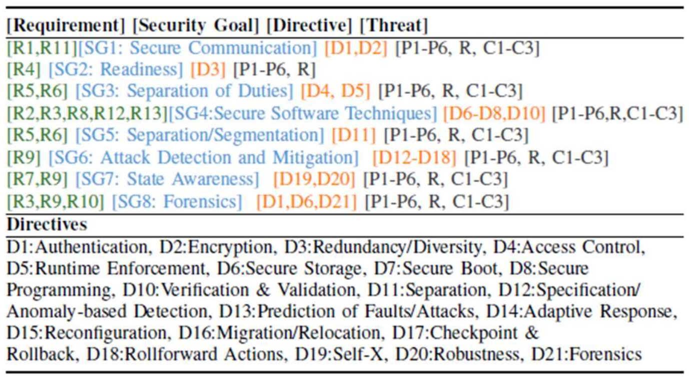
In this paper, we have identified general requirements to ensure a secure software update process. These requirements fulfill common security goals for cyber-secure vehicles. Moreover, we present a reference architecture named UniSUF based on previous work. We validate the usability and security of our reference architecture by identifying an attacker model and performing a threat assessment. Finally, we identify mitigation mechanisms and map the specific threats to security goals and requirements to strengthen the robustness and design of UniSUF for a broad industry acceptance with UN Regulation No. 156 in mind.

We assume a common agenda where someone aims to manipulate the software update process or the software itself at any entity or during communication between entities throughout the software update process. For instance, the intent can be to recover and exploit secret signing or obtain encryption keys used during the software update process. The latter might enable disabling firewalls or switching ECUs into programming mode to enable update capabilities. Additionally, attackers might want to decrypt software files to reverse engineer and gain insight into its contents affecting the intellectual property and try to find vulnerabilities, e.g., through analysis of safety-critical systems. Thus, the attacker’s ultimate goal is to exploit the software update system so that malicious or unauthorized software providing additional or altered functionality reaches the in-vehicle system, for instance, to gain and maintain remote persistence.

Thirteen security requirements were identified which are mapped against 8 security goals and 21 security directives. We further use Goal Structuring Notation (GSN) to present proofs for claims in a graphical manner to map these claims to the general requirements, as illustrated in the following figure:



Mapping of Requirements to Security Goals, Directives and Threats



[Requirement]	[Security Goal]	[Directive]	[Threat]
[R1,R11]	[SG1: Secure Communication]	[D1,D2]	[P1-P6, R, C1-C3]
[R4]	[SG2: Readiness]	[D3]	[P1-P6, R]
[R5,R6]	[SG3: Separation of Duties]	[D4, D5]	[P1-P6, R, C1-C3]
[R2,R3,R8,R12,R13]	[SG4:Secure Software Techniques]	[D6-D8,D10]	[P1-P6,R,C1-C3]
[R5,R6]	[SG5: Separation/Segmentation]	[D11]	[P1-P6, R, C1-C3]
[R9]	[SG6: Attack Detection and Mitigation]	[D12-D18]	[P1-P6, R, C1-C3]
[R7,R9]	[SG7: State Awareness]	[D19,D20]	[P1-P6, R, C1-C3]
[R3,R9,R10]	[SG8: Forensics]	[D1,D6,D21]	[P1-P6, R, C1-C3]

Directives

D1:Authentication, D2:Encryption, D3:Redundancy/Diversity, D4:Access Control, D5:Runtime Enforcement, D6:Secure Storage, D7:Secure Boot, D8:Secure Programming, D10:Verification & Validation, D11:Separation, D12:Specification/ Anomaly-based Detection, D13:Prediction of Faults/Attacks, D14:Adaptive Response, D15:Reconfiguration, D16:Migration/Relocation, D17:Checkpoint & Rollback, D18:Rollforward Actions, D19:Self-X, D20:Robustness, D21:Forensics

For instance, Security Goal **SG1**, **Secure communication**, is fulfilled by requirements R1 and R11. Requirement **R1** deals with ***infrastructure and communication***: "The infrastructure, cryptographic algorithms, and key material shall follow best security practices. For instance, communication between backend entities shall encrypt communication and use proper authentication between entities. The same requirements shall be considered for in-vehicle entities directly related to the update framework." R1 should then be reinforced by implementing the detailed directives **D1 Authentication** and **D2 Encryption**, mitigating threats P1-P6, R, and C1-C3 (not shown here in these figures).

More details and how packet validation and cryptographic material is derived and used, can be found in the paper.

EXTRACT FROM PUBLICATION

Abstract – A modern vehicle is no longer merely a transportation vessel. It has become a complex cyber-physical system containing over 100M lines of software code controlling various functionalities such as safety-critical steering, brake, and engine control. The amount of code is anticipated to rise to around 300M lines of code by 2030. Furthermore, even well-tested code will contain more than one bug per 1000 lines of code. Thus, it can be expected that there will be around 100k bugs in a modern vehicle and around 300k bugs in a few years, where some might have a safety-critical impact. Automotive companies are transforming into software companies with more software developed in-house. The ability to patch vulnerabilities hastily and securely has become vital and is a prerequisite when securing modern cars. UN Regulation No. 156 and the ISO 24089 emphasize the ability to update vehicle software securely.

Consequently, we focus on securing the vehicle software update process. Our contributions include defining an attacker model and general security requirements. We further map these requirements to common security goals and directives to ensure broad coverage. Additionally, we present UniSUF, a secure and versatile approach to vehicle software updates. We identify entities involved during vehicle software updates, perform a threat assessment, and map the identified threats to security goals and requirements. The results highlight a secure framework with high industrial relevance that can be used as a reference architecture to guide securing similar software update systems within automotive and related areas such as cyber-physical systems, internet-of-things, and smart cities.

Conclusions – Modern vehicles are complex systems containing more than 100M lines of software code controlling various functionality including safety-critical functions and get increasingly vulnerable when adding connectivity. Thus, ensuring hastily and secure software updates to patch vulnerabilities is imperative. We have introduced UniSUF, identified entities involved in the distribution and execution during vehicle software updates, provided an attacker model, performed a threat assessment, and elaborated on mitigation mechanisms. We have identified general security requirements for vehicle software updates and mapped them to common security goals and directives further visualized with the Goal Structuring Notation (GSN). The results show that UniSUF fulfills the stated security goals and provides a secure and unified vehicle software update framework that can serve as a detailed reference architecture. We believe our results are valuable not only for automotive software update architects. We also see high relevance for engineers in related areas, such as cyber-physical systems, internet-of-things, and smart cities, guiding the design of secure software update solutions.

7.2 UniSUF: a unified software update framework for vehicles utilizing isolation techniques and trusted execution environments [26]

Paper presented at the 19th ESCAR Europe conference 2021, pp. 86-100.

Authors: K. Strandberg, D Kengo Oka, T. Olovsson

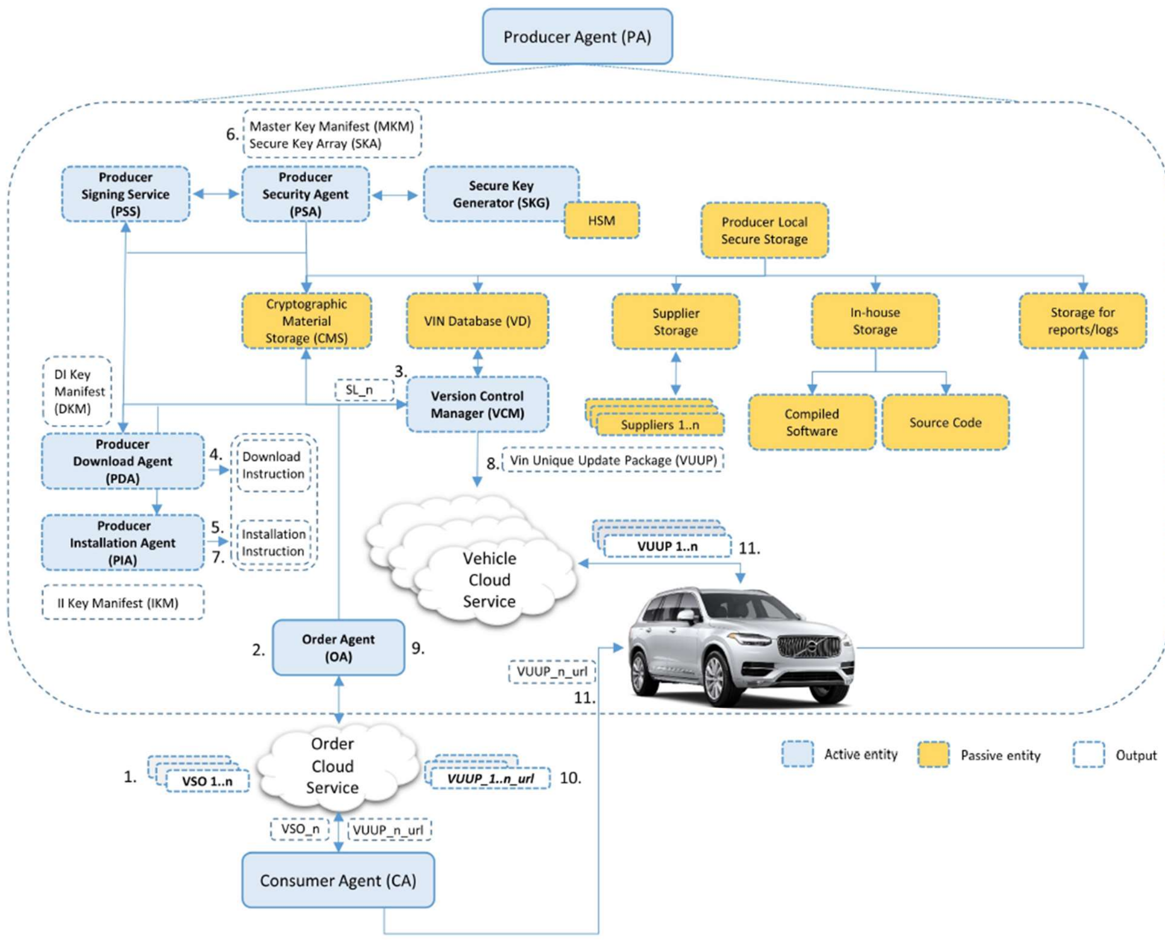
The complete text is available after registration at https://www.escar.info/history/escar-europe/escar-europe-2021-lectures-and-program-committee.html

Considering the different existing use cases for vehicle software updates, such as over-the-air, using a workshop computer, at factory production, or with a diagnostic update tool, each use case typically has its own approach which causes complexity. Moreover, new use cases for software updates need to be considered with future demands to support 3rd party component updates. Therefore, to simplify, reduce costs, allow flexibility, and to make the update process manageable, all while considering security aspects, we propose a unified and versatile approach to handle all listed/mentioned use cases.

After reviewing the above-mentioned use cases, the following constraints and conditions are defined for a unified software update framework:

- Support for online updates (software update files and/or cryptographic credentials/operations require online access). UniSUF: A Unified Software Update Framework for Vehicles.
- Support for offline updates (software update files and cryptographic credentials/operations are accessible offline).
- Should not rely on additional input for cryptographic keys or installation instructions, e.g., from a diagnostic update tool (i.e., all data needed for a complete software update is securely encapsulated into one single file and no additional input is required).
- No dependency on the data distribution model (i.e., software update files can be provided through different means, and it does not matter how they are distributed to the vehicle).
- No dependency on software update storage location (i.e., software update files should be independently protected regardless of where they are stored).
- Flexible and modular to support 3rd party component updates. We have taken these constraints and conditions into consideration when designing a software update framework to allow for a unified and versatile approach to support different use cases.

Both data handling, signing, and verification of update package contents in the backend system and the data distribution to the vehicle is considered. The solution is rather complex although reasonably simple to understand its main functionality and how the different components are tied together. The figure below shows data distribution in the backend:



More details and how packet validation and cryptographic material is derived and used, can be found in the paper.

EXTRACT FROM PUBLICATION

Abstract – Today's vehicles depend more and more on software and can contain over 100M lines of code controlling many safety-critical functions, such as steering and brakes. Increased complexity in software inherently increases the number of bugs affecting vehicle safety-critical functions. Consequently, software updates need to be applied regularly. Current research around vehicle software update solutions is lacking necessary details for a versatile, unified and secure approach that covers various update scenarios, e.g., over-the-air, with a workshop computer, at factory production or using a diagnostic update tool. We propose UniSUF, a Unified Software Update Framework for Vehicles, well aligned with automotive industry stakeholders. All data needed for a complete software update is securely encapsulated into one single file. This vehicle-unique file can be processed in multitudes of update scenarios and executed without any external connectivity since all data is inherently secured. To the best of our knowledge, this comprehensive, versatile, and unified approach cannot be found in previous research and is a contribution to an essential requirement within the industry for handling the increasing complexity related to vehicle software updates.

Conclusions – UniSUF is made to accommodate various scenarios for the automotive domain by encapsulating needed data into one single file, a Vehicle Unique Update Package (VUUP). This vehicle-unique file can be processed within a vehicle ECU, using a workshop computer, at factory production, with a diagnostic update tool, or in other compositions. Moreover, the complete update process can be performed without any external communication dependencies since all files are inherently secured. A continuous secure software update process is a prerequisite for facilitating vehicle resilience towards cyberattacks in a rapidly changing environment. We believe our contributions in this paper can facilitate further research in this area, towards securing the connected car.

8 Endurance, verification and validation

Detection of unwanted activities and deviation from normal behavior may be detected by regular IDS systems. Trusted components can be used to evaluate functionality and behavior and it can make IDS systems hard to compromise. In addition, cloud-based IDS systems may be deployed and protect complete fleets and prevent problems from spreading. It is not obvious that a vehicle can or should be trusted to evaluate its own state and whether it is fully functional. If it is compromised, internal IDS systems may be failing to see the problem due to the sophistication of an attack, or the IDS system(s) may even be compromised as well. We have therefore taken a different approach to this problem to make sure misbehaving and compromised vehicles are detected and either fixed or removed from traffic. The methodology is based on peer assessment of vehicles, where vehicles assess each other after they have been interacting with each other and upload their verdicts to the cloud. This approach makes it possible to verify vehicle behavior and detect changes over a longer time and to follow each vehicle's behavior. Changes do not necessarily have to do with cyber-attacks, but the system will also react to changes in behavior due to changes in vehicle surroundings that may call for software or hardware updates.

8.1 V2C: A Trust-Based Vehicle to Cloud Anomaly Detection Framework for Automotive Systems [8]

Paper presented at the 16th International Conference on Availability, Reliability and Security ARES 2021. New York, NY, USA.

Authors: T. Rosenstatter, T. Olovsson, and M. Almgren

The complete text is available online and is not duplicated here for copyright reasons.

Vehicles know quite a lot about each other by observing their behavior, and after interacting with another vehicle, for example in an intersection or after having participated in a platoon, they can give a verdict about the other vehicles by assigning them trust scores. If a vehicle misbehaves, for example transmits a speed that is obviously wrong, a position that does not match what other vehicles observe, or if it in any other way behaves unexpectedly or aggressively, the trust score indicates this. There are many scenarios that can be covered by this approach:

Scenario 1 – Unauthorized firmware manipulation:

> Ex.1 Manipulation of the firmware such that the owner is able to send traffic congestion warning messages at will to reduce traffic on a desired road segment.
>
> Ex.2 Manipulation of the firmware such that the automated vehicle drives faster than the current speed limit. Ex.3 Manipulation of the firmware such that an attacker can disrupt traffic by suppressing relaying of messages, spoofing warning messages or flooding the VANET with erroneous messages.

Scenario 2 – HW/SW failures:

> Ex.4 A lidar sensor or camera is experiencing a fault and the vehicle is thus not able to perceive its environment properly.

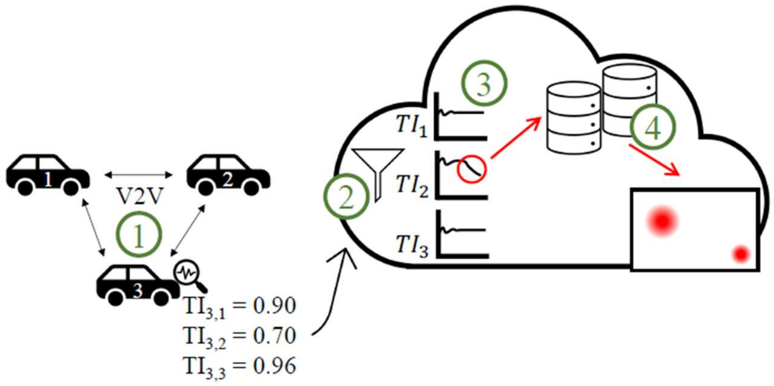
Scenario 3 – Legitimate SW/HW updates:

Ex.5 After a legitimate firmware update, the automated vehicle drives too fast in certain road conditions, e. g., when the road is slippery, since the vehicle perceives the current driving conditions incorrectly.

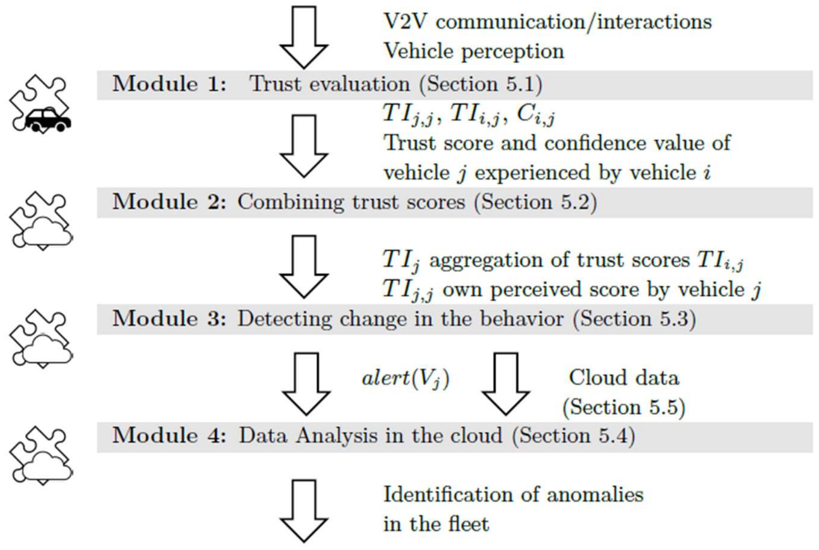
Ex.6 A defective hardware component is replaced in an authorized workshop and causes compatibility problems resulting in a misbehavior while driving.

Ex.7 The machine learning algorithm for identifying traffic signs has been updated and now causes a misclassification of speed limit signs.

The figure below shows three vehicles interacting with each other where vehicle 3 assigns a trust score for all three:



These trust scores are then uploaded to the cloud for analysis, and if many vehicles report low trust scores for a vehicle, it should be subject to a closer investigation for example by the OEM. The proposed system consists of four modules (or four steps) for how the analysis is done:


V2V communication/interactions
Vehicle perception
Module 1: Trust evaluation (Section 5.1)
$TI_{j,j}$, $TI_{i,j}$, $C_{i,j}$
Trust score and confidence value of vehicle j experienced by vehicle i
Module 2: Combining trust scores (Section 5.2)
TI_j aggregation of trust scores $TI_{i,j}$
$TI_{j,j}$ own perceived score by vehicle j
Module 3: Detecting change in the behavior (Section 5.3)
$alert(V_j)$
Cloud data (Section 5.5)
Module 4: Data Analysis in the cloud (Section 5.4)
Identification of anomalies in the fleet

Vehicle trust is primarily proposed in literature to overcome the difficulty of relying on information received via V2V communication when interacting with other automated vehicles. Related work also identified the potential of vehicle trust in combination with intrusion detection, however, these solutions are either focusing on collaborating with trusted vehicles and including the results in the own vehicle's decision-making [22] or only consider packet header information in their intrusion detection system [23][24].

We propose an anomaly detection framework that utilizes the trust scores computed by individual vehicles based on V2V interactions by combining them with a subsequent analysis in the cloud. The presented V2C Anomaly Detection framework consists of four modules dividing the tasks into (i) individual assessments of neighboring vehicles resulting in a trust score; (ii) aggregation of these individual assessments to one trust score per vehicle; (iii) detection of anomalies or changes in the trust score over time; (iv) further analysis using data available in the cloud to also detect similar anomalies in a vehicle. The advantage of using trust evaluations for detecting anomalies is twofold. First, these vehicle assessments are, unlike IDSs, performed by other vehicles and not by the system itself, which is important because the vehicle itself may not be aware of the fault or the IDS may also be compromised.

Second, this framework is scalable as the computational costs for observing the trust score for each vehicle and triggering a detailed analysis only when changes in the score are detected, requires less resources than triggering a comprehensive analysis with each newly uploaded event to the cloud. For each module comprising this framework we define their requirements, show how the identified threats can be detected, provide detailed discussions about suitable techniques, and propose modifications if necessary. Furthermore, we identify attack scenarios which such a framework can detect and discuss its applicability in a detailed discussion based on a use case.

EXTRACT FROM PUBLICATION

ABSTRACT – Vehicles have become connected in many ways. They communicate with the cloud and will use Vehicle-to-Everything (V2X) communication to exchange warning messages and perform cooperative actions such as platooning. Vehicles have already been attacked and will become even more attractive targets due to their increasing connectivity, the amount of data they produce and their importance to our society. It is therefore crucial to provide cyber security measures to prevent and limit the impact of attacks.

As it is problematic for a vehicle to reliably assess its own state when it is compromised, we investigate how vehicle trust can be used to identify compromised vehicles and how fleet-wide attacks can be detected at an early stage using cloud data. In our proposed V2C Anomaly Detection framework, peer vehicles assess each other based on their perceived behavior in traffic and V2X-enabled interactions and upload these assessments to the cloud for analysis. This framework consists of four modules. For each module we define functional demands, interfaces and evaluate solutions proposed in literature allowing manufacturers and fleet owners to choose appropriate techniques. We detail attack scenarios where this type of framework is particularly useful in detecting and identifying potential attacks and failing software and hardware. Furthermore, we describe what basic vehicle data the cloud analysis can be based upon.

CONCLUSION – In this paper, we present the V2C Anomaly Detection framework, which is a novel framework combining the assessment of Vehicle-to-Vehicle communication and the perceived quality of cooperative interactions between vehicles resulting in a trust score, with vehicle data in the cloud. The peer evaluation of vehicle behavior allows identification of local anomalies and attacks even when important security controls such as in-vehicle IDSs fail to detect them, for example due to an attacker exploiting a vulnerability, an insider or a firmware upgrade causing unintended behavior. Furthermore, the analysis of cloud data makes it possible to detect and identify patterns of anomalies and intrusions on a wider scale such as on a fleet level. Ultimately, the advantage of the

V2C Anomaly Detection framework lies in the fact that it is designed to reduce the computational costs in the cloud by triggering a cloud analysis once the combined trust evaluation performed by independent vehicles shows a significant change, i. e., a changed behavior resulting in a decline of the trust score.

We have provided scenarios focusing on persistent threats to explain the requirements for each module of the V2C Anomaly Detection framework in terms of functionality, inputs, and outputs. We also provide an initial identification and detailed discussions to aid in choosing or adapting techniques for each module so that a vehicle manufacturer, fleet owner or other actor in the cloud can select and adapt relevant techniques depending on the available data.

9 Performance of communication

Reliable communication between vehicles and the infrastructure is essential for many functions. We have studied byzantine faults (or failures) in communication where components may silently fail without this being recognized by others. It affects for example consensus protocols and algorithms where two or more components all need to be in a known state. It is essential in many decisions made by communicating vehicles (V2X communication), but also inside a vehicle where decisions need to be made in a consistent way. Secure software updates can be one special case, where all or no ECUs should be updated to the same known state. Other examples are fault tolerant components in the vehicle which may fail and require a, for the driver, invisible reconfiguration and reassignment of functions between ECUs.

Fault-tolerant distributed systems are known to be hard to design and verify. High-level communication primitives can facilitate such complex challenges. These primitives can be based on low-level ones, e.g., the one that allows processes to send a message to only one other process at a time. Hence, when an algorithm wishes to broadcast message m to all processes (or ECUs), it can send m individually to every other process. But, if the sender fails during this broadcast, perhaps only some of the processes have received m. Even in the presence of network level support for broadcasting or multicasting, failures can cause similar inconsistencies. To simplify the design of fault-tolerant distributed algorithms, such inconsistencies need to be avoided. Fault-tolerant broadcasts can simplify the development of fault-tolerant distributed systems, e.g., State Machine Replication and Set-Constrained Delivery Broadcast. The weakest variant, Reliable Broadcast, lets all non-failing processes agree on the set of delivered messages, including all messages they have broadcast. We aim to design a reliable broadcast that is more fault-tolerant than the current state-of-the-art.

We have studied a well-known communication abstraction called Byzantine Reliable Broadcast (BRB). This abstraction is central in the design and implementation of fault-tolerant distributed systems, as many fault-tolerant distributed applications require communication with provable guarantees on message deliveries. Our study focuses on fault-tolerant implementations for message-passing systems that are prone to process failures, such as crashes and malicious behaviors.

9.1 Self-stabilizing Byzantine-Tolerant Recycling [30]

Paper presented at the 25th International Symposium on Stabilization, Safety, and Security of Distributed Systems, SSS 2023, Jersey City, USA.

Authors: C. Georgiou, M. Raynal, E. Schiller

The complete text is available online and is not duplicated here for copyright reasons.

Abstract – Numerous distributed applications, such as cloud computing and distributed ledgers, necessitate the system to invoke asynchronous consensus objects for an unbounded number of times, where the completion of one consensus instance is followed by the invocation of another. With only a constant number of objects available, object reuse becomes vital. We investigate the challenge of object recycling in the presence of Byzantine processes, which can deviate from the algorithm code in any manner. Our solution must also be self-stabilizing, as it is a powerful notion of fault tolerance. Self-stabilizing systems can recover automatically after the occurrence of arbitrary

transient faults, in addition to tolerating communication and (Byzantine or crash) process failures, provided the algorithm code remains intact. We provide a recycling mechanism for asynchronous objects that enables their reuse once their task has ended, and all non-faulty processes have retrieved the decided values. This mechanism relies on synchrony assumptions and builds on a new self-stabilizing Byzantine-tolerant synchronous multivalued consensus algorithm, along with a novel composition of existing techniques.

Conclusions – We have presented an SSBFT algorithm for object recycling. Our proposal can support an unbounded sequence of SSBFT object instances. The expected stabilization time is in O(t) synchronous rounds. We believe that this work is preparing the groundwork needed to construct SSBFT Blockchains. As a potential avenue for future research, one could explore deterministic recycling mechanisms, say by utilizing the Dolev and Welch approach to SSBFT clock synchronization, to design an SSBFT SIG-index. However, their solution has exponential stabilization time, making it unfeasible in practice.

9.2 Self-stabilizing Byzantine Fault-Tolerant Repeated Reliable Broadcast [29]

Paper published in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 13751 LNCS s. 206-221, 9783031210167 (ISBN).

Authors: R. Duvignau, M. Raynal, E. Schiller

The complete text is available online and is not duplicated here for copyright reasons.

Abstract – We study a well-known communication abstraction called Byzantine Reliable Broadcast (BRB). This abstraction is central in the design and implementation of fault-tolerant distributed systems, as many fault-tolerant distributed applications require communication with provable guarantees on message deliveries. Our study focuses on fault-tolerant implementations for message-passing systems that are prone to process failures, such as crashes and malicious behaviors.

At PODC 1983, Bracha and Toueg, in short, BT, solved the BRB problem. BT has optimal resilience since it can deal with up to $t < n/3$ Byzantine processes, where n is the number of processes. The present work aims at the design of an even more robust solution than BT by expanding its fault-model with self-stabilization, a vigorous notion of fault-tolerance. In addition to tolerating Byzantine and communication failures, self-stabilizing systems can recover after the occurrence of arbitrary transient faults. These faults represent any violation of the assumptions according to which the system was designed to operate (as long as the algorithm code remains intact).

We propose, to the best of our knowledge, the first self-stabilizing Byzantine fault-tolerant (SSBFT) solution for repeated BRB (that follows BT's specifications) in signature-free message-passing systems. Our contribution includes a self-stabilizing variation on a BT that solves asynchronous single-instance BRB. We also consider the problem of recycling instances of single-instance BRB. Our SSBFT recycling for time-free systems facilitates the concurrent handling of a predefined number of BRB invocations and, this way, can serve as the basis for SSBFT consensus.

Conclusions – To the best of our knowledge, this paper presents the first SSBFT algorithms for IRC and repeated BRB for hybrid asynchronous/timefree systems. As in BT, the SSBFT BRB algorithm takes several asynchronous communication rounds of O(n2) messages per instance whereas the IRC algorithm takes O(n) messages but requires synchrony assumptions.

The two SSBFT algorithms are integrated via specified interfaces and message piggybacking. Thus, our SSBFT repeated BRB solution increases BT's message size only by a constant per BRB, but the number of messages per instance stays similar. The integrated solution can run an unbounded number of (concurrent and independent) BRB instances. The advantage is that the more communication-intensive component, i.e., SSBFT BRB, is not associated with any synchrony assumption. Specifically, one can run δ concurrent BRB instances, where δ is a parameter for balancing the trade-off between fault recovery time and the number of BRB instances that can be used (before the next δ concurrent instances can start). The above extension mitigates the effect of the fact that, for the repeated BRB problem, muteness detectors are used and mild synchrony assumptions are made in order to circumvent well-known impossibilities. Those additional assumptions are required for the entire integrated solution to work. To the best of our knowledge, there is no proposal for a weaker set of assumptions for solving the studied problem in a self-stabilizing manner.

We hope that the proposed solutions, e.g., the proposed recycling mechanism and the hybrid composition of time-free/asynchronous system settings, will facilitate new SSBFT building blocks.

9.3 Brief Announcement: Self-stabilizing Total-Order Broadcast [28]

Paper published in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 13751 LNCS s. 358-363, 9783031210167 (ISBN).

Authors: O. Lundström, M. Raynal, E. Schiller

The complete text is available online and is not duplicated here for copyright reasons.

Abstract – Our study aims at the design of an even more reliable solution. We do so through the lenses of self-stabilization—a very strong notion of fault-tolerance. In addition to node and communication failures, self-stabilizing algorithms can recover after the occurrence of arbitrary transient faults; these faults represent any violation of the assumptions according to which the system was designed to operate (as long as the algorithm code stays intact). This work proposes the first (to the best of our knowledge) self-stabilizing algorithm for total-order (uniform reliable) broadcast for asynchronous message-passing systems prone to process failures and transient faults. As we show, the proposed solution facilitates the elegant construction of self-stabilizing state-machine replication using bounded memory.

Discussion – Our study aims at the design of an even more reliable solution. We do so through the lenses of self-stabilization—a very strong notion of fault-tolerance. In addition to node and communication failures, self-stabilizing algorithms can recover after the occurrence of arbitrary transient faults; these faults represent any violation of the assumptions according to which the system was designed to operate (as long as the algorithm code stays intact). This work proposes the first (to the best of our knowledge) self-stabilizing algorithm for total-order (uniform reliable) broadcast for asynchronous message-passing systems prone to process failures and transient faults. As we show, the proposed solution facilitates the elegant construction of self-stabilizing state-machine replication using bounded memory.

10 References

[1] ISO, “ISO 26262:2011 Road vehicles – functional safety,” International Organization for Standardization, Standard, 2011.

[2] ISO/TR 4804:2020, Road vehicles — Safety and cybersecurity for automated driving systems — Design, verification and validation. International Organization for Standardization, Standard, 2020.

[3] T. Rosenstatter, “On the Secure and Resilient Design of Connected Vehicles: Methods and Guidelines”. Ph.D. thesis, Chalmers University of Technology, 2021. https://research.chalmers.se/publication/526019

[4] K. Tuma, “Efficiency and Automation in Threat Analysis of Software Systems”. Ph.D thesis, Gothenburg University 2021. https://research.chalmers.se/publication/520907

[5] K. Strandberg, “Towards a Secure and Resilient Vehicle Design: Methodologies, Principles and Guidelines”, Licentiate thesis, Chalmers University of Technology, https://research.chalmers.se/publication/529239

[6] T. Rosenstatter, C. Englund (2017) “Modelling the Level of Trust in a Cooperative Automated Vehicle Control System”. IEEE Transactions on Intelligent Transportation Systems, 19(4) pp. 1237-1247.

[7] T. Rosenstatter, K. Strandberg, R. Jolak, R. Scandariato, T. Olovsson. “REMIND: A Framework for the Resilient Design of Automotive Systems”, 2020 IEEE Secure Development (SecDev), 2020, p. 81-95

[8] T. Rosenstatter, T. Olovsson, M. Almgren, “V2C: A Trust-Based Vehicle to Cloud Anomaly Detection Framework for Automotive Systems”, Proceedings of the 16th International Conference on Availability, Reliability and Security (ARES 2021), 2021, p. 1-10

[9] K. Strandberg, T. Rosenstatter, R. Jolak, N. Nowdehi and T. Olovsson, "Resilient Shield: Reinforcing the Resilience of Vehicles Against Security Threats," 2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring), 2021, pp. 1-7, doi: 10.1109/VTC2021-Spring 51267.2021.9449029.

[10] D. Dubrefjord, M. Jang, H. Hadi, T. Olovsson, “Security of In-Vehicle Communication Systems”. Book chapter in Decision Support Systems and Industrial IoT in Smart Grid, Factories, and Cities, 2021, p. 162-179, ISBN: 9781799874683

[11] M. Folkemark, V. Rydberg, “Performance Evaluation of a Hardware Security Module in Vehicles”, Master Thesis work at Chalmers University of Technology, 2021.

[12] W. Aoudi, M. Almgren, N. Nowdehi, T. Olovsson, “Spectra: Detecting Attacks on In-Vehicle Networks through Spectral Analysis of CAN-Message Payloads”, Proceedings of the ACM Symposium on Applied Computing, SAC ’21, March 22–26, 2021, pp. 1588-1597, ISBN 9781450381048

[13] A. Avizienis, J. . Laprie, B. Randell, and C. Landwehr, “Basic concepts and taxonomy of dependable and secure computing,” IEEE Transactions on Dependable and Secure Computing, vol. 1, no. 1, pp. 11–33, 2004.

[14] J.-C. Laprie, “From dependability to resilience,” in 38th IEEE/IFIP Int. Conf. On Dependable Systems and Networks, 2008, pp. G8–G9.

[15] D. Ratasich, F. Khalid, F. Geissler, R. Grosu, M. Shafique, and E. Bartocci, “A roadmap toward the resilient internet of things for cyber-physical systems,” IEEE Access, vol. 7, pp. 13 260–13 283, 2019.

[16] S. Hukerikar and C. Engelmann, “Resilience design patterns: A structured approach to resilience at extreme scale,” arXiv preprint arXiv:1708.07422, 2017.

[17] V. Chang, M. Ramachandran, Y. Yao, Y.-H. Kuo, and C.-S. Li, “A resiliency framework for an enterprise cloud,” International Journal of Information Management, vol. 36, no. 1, pp. 155 – 166, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S026840121500095X

[18] Microsoft Corporation, “The STRIDE threat model,” 2005, (Accessed: 2022-01-21). [Online]. Available: https://msdn.microsoft.com/en-us/library/ee823878.aspx

[19] M. M. Islam, A. Lautenbach, C. Sandberg, and T. Olovsson, “A risk assessment framework for automotive embedded systems,” in Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security - CPSS 16. Association for Computing Machinery (ACM), 2016.

[20] B. Sangchoolie, P. Folkesson, P. Kleberger and J. Vinter, "Analysis of Cybersecurity Mechanisms with respect to Dependability and Security Attributes," 2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W), 2020, pp. 94-101.

[21] F. Sommer, J. Dürrwang, and R. Kriesten, “Survey and classification of automotive security attacks,” Information, vol. 10, no. 4, 2019. [Online]. Available: https://www.mdpi.com/2078-2489/10/4/148

[22] P. Guo, H. Kim, L. Guan, M. Zhu, and P. Liu, “VCIDS: Collaborative intrusion detection of sensor and actuator attacks on connected vehicles,” in Security and Privacy in Communication Networks, X. Lin, A. Ghorbani, K. Ren, S. Zhu, and A. Zhang, Eds. Cham: Springer International Publishing, 2018, pp. 377–396.

[23] T. Nandy, R. M. Noor, M. Yamani Idna Bin Idris, and S. Bhattacharyya, “TBCIDS: Trust-based collaborative intrusion detection system for VANET,” in 2020 National Conference on Emerging Trends on Sustainable Technology and Engineering Applications (NCETSTEA), Durgapur, India, 2020, pp. 1–5.

[24] E. A. Shams, A. Rizaner, and A. H. Ulusoy, “Trust aware support vector machine intrusion detection and prevention system in vehicular ad hoc networks,” Computers & Security, vol. 78, pp. 245–254, 2018.

[25] I. Loskin: “TARA+AD: Threat Analysis and Risk Assessment for Automated Driving: cybersecurity of road vehicles”, University of Juväskylä, Finland, 2023. https://jyx.jyu.fi/handle/123456789/87888

[26] K. Strandberg, D. Kengo Oka, T. Olovsson, ” UniSUF: A unified software update framework for vehicles utilizing isolation techniques and trusted execution environments”, 19th ESCAR Europe conference 2021, pp. 86-100, https://www.escar.info/history/escar-europe/escar-europe-2021-lectures-and-program-committee.html

[27] A. Lautenbach, M. Almgren, T. Olovsson: ” Proposing HEAVENS 2.0 – an automotive risk assessment model”, Proceedings - Computer Science in Cars Symposium (CSCS ’21): ACM Computer Science in Cars Symposium, 9781450391399 (ISBN)

[28] O. Lundström, M. Raynal, E. Schiller, ” Brief Announcement: Self-stabilizing Total-Order Broadcast”, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 13751 LNCS s. 358-363, 9783031210167 (ISBN)

[29] R. Duvignau, M. Raynal, E. Schiller, “Self-stabilizing Byzantine Fault-Tolerant Repeated Reliable Broadcast”, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 13751 LNCS s. 206-221, 9783031210167 (ISBN)

[30] C. Georgiou, M. Raynal, E. Schiller: “Self-stabilizing Byzantine-Tolerant Recycling”. 25th International Symposium on Stabilization, Safety, and Security of Distributed Systems, SSS 2023, Jersey City, USA.

[31] D. Grimm, A. Lautenbach, M. Almgren, T. Olovsson, ” Gap analysis of ISO/SAE 21434 – Improving the automotive cybersecurity engineering life cycle”, 2023 IEEE 26th International Conference on Intelligent Transportation Systems, ITSC, 2023,

[32] K. Strandberg, U. Arnljung, T. Olovsson, D. Kengo Oka, ” Secure Vehicle Software Updates: Requirements for a Reference Architecture”, IEEE Vehicular Technology Conference VTC-2023, 979-8-3503-1114-3 (ISBN)

[33] K. Strandberg, N. Nowdehi, T. Olovsson, ”A Systematic Literature Review on Automotive Digital Forensics: Challenges, Technical Solutions and Data Collection”, IEEE Transactions on Intelligent Vehicles, 23798858 (eISSN) Vol. 8, pp 1350-1367

[34] K. Strandberg, U. Arnljung, and T. Olovsson. “The Automotive BlackBox: Towards a Standardization of Automotive Digital Forensics”. In: IEEE International Workshop on Information Forensics and Security (2023)