Extending Software Architectures from Safety to Security

Martin Böhner, Alexander Mattausch, Alexander Much
CIS-Technology
Elektrobit Automotive GmbH
Am Wolfsmantel 46
91058 Erlangen
martin.boehner@elektrobit.com
alexander.mattausch@elektrobit.com
alexander.much@elektrobit.com

Abstract: In this paper we summarize approaches for software architectures used in the automotive domain for safety-critical or mixed safety-critical systems and extend the approach to security-critical systems. Safety and security aspects of systems influence each other and we show solutions which combine both worlds in a common architectural and development process approach.

1 Motivation and Background

In recent years several studies (e.g. [KCR+10], [MV13], [Dan10]) revealed a growing security threat potential for cars from all kinds of aspects, e.g.

- System and software complexity: e.g. functionality per ECU, lines of code
- Connectivity: e.g. WiFi, Bluetooth, Ethernet and Internet, Car2Car, Car2X
- Interoperability: e.g. WiFi, Bluetooth, Internet, mobile phone integration
- Sped up time-to-market due to the necessity to adapt quickly: e.g. Apps, 3rd party SW

For future applications in the automotive domain the topics connectivity and collaboration with various outside sources are essential features. On the one hand this includes communication with various sensors and applications in the car itself. On the other hand, however, this will also comprise external sources such as mobile devices, roadside infrastructure or backend IT systems that provide cloud computing capacities.

The signals exchanged here will be utilized in safety and security critical use cases like assisted and autonomous driving, or in the exchange of traffic and hazard information between cars and roadside infrastructure.

As shown by different independent studies the mutual influence between safety and security aspects is very strong. Safety software engineering and the resulting applications

Figure 1: Connected car with various communication partners

have a direct impact on security and vice versa. This has been recognized by established software safety standards like IEC 61508 [IEC10] or ISO 26262 [ISO11b]. And it has also been addressed by regulators of other domains, e.g. by the FDA jointly with the ICS-CERT in 2013: *"Manufacturers are responsible for remaining vigilant about identifying risks and hazards associated with their medical devices, including risks related to cybersecurity, and are responsible for putting appropriate mitigations in place to address patient safety and assure proper device performance"* [FDA13], [IC13]. Even regulatory bodies or political institutions like the European Commission correlate both topics, e.g. José Manuel Durão Barroso in a speech about nuclear safety in 2012: *"[. . .] there is no safety without security and vice-versa"*, see [Jos12].

There are already initiatives by standardization bodies to identify synergies in creating the assurance cases that combine approaches like safety cases, security assurance and system integrity levels, e.g. ISO 15026 [ISO11a].

A practical consequence for today's systems is that most safety critical use cases will have to consider intentional manipulation of messages and data as well as direct attacks on software and hardware level. Such systems need to be accompanied with a state of the art argumentation for safety and security, see e.g. the BGH airbag case [Bun09]. However, the practical interpretation of "state of the art" is changing quickly and system development needs to take this background into account.

This article presents practical approaches for software architectures addressing the integrity of automotive ECUs using established environments like AUTOSAR (see [AUT14a]). Such software architectures provide methods to address safety, security and availability aspects of today's state of the art ECUs.

2 Integrity and Safety Mechanisms

The basic software architecture of a safety-critical system is responsible for providing mechanisms to establish the integrity of the system from a software point of view. Such mechanisms also include the critical interfaces between hardware and software, including detection mechanisms of hardware faults. AUTOSAR provides a framework that allows Software Components (SW-Cs), as the building blocks of a full AUTOSAR application are called, to concentrate on safety-critical functions independent of the practical implementation of the basic software layers.

The provision of such basic mechanisms is well established, e.g. in the definitions of "independence" in IEC 61508 [IEC10] or *freedom from interference* or *criteria for coexistence of elements* in ISO 26262 [ISO11b]. Technically the "best" definition can be found in DO-178C (see [RTC11], 2.4). It defines three classical architectural considerations: partitioning, multi-version dissimilar software and safety monitoring.

This chapter mentions these methods briefly as reference. The focus of this paper is on the extension of these safety mechanisms to security mechanisms. Further details can be found in [MA09], [HJMA14], [GM18].

2.1 Memory Partitioning

In most software architectures memory partitioning is provided by the operating system. For safety-critical systems this includes at least memory write protection to separate different tasks from each other. State of the art operating systems also strictly separate the operating system kernel from all other parts of the software system. This usually results in a software architecture with a clear separation of safety monitors from the monitored safety functions as well as a separation of multi-version dissimilar software.

State of the art operating systems also provide stack protection features to prevent different tasks from writing to stacks of other tasks and to prevent or at least limit stack overruns. Additionally, the execution context of each task is encapsulated in the operating system to prevent write access from other tasks.

Memory partitioning is based on hardware features such as memory protection units (MPU) or memory management units (MMU). The level of protection provided by the operating system varies with the different hardware implementations.

See Figure 2 for a sample software architecture including protection mechanism for safety-critical ECUs using the AUTOSAR software architecture as basis.

2.2 Execution and Scheduling Protection

Independence in software has two aspects: spatial and temporal independence. Memory protection and a good analysis of the hardware-software interactions including reactions

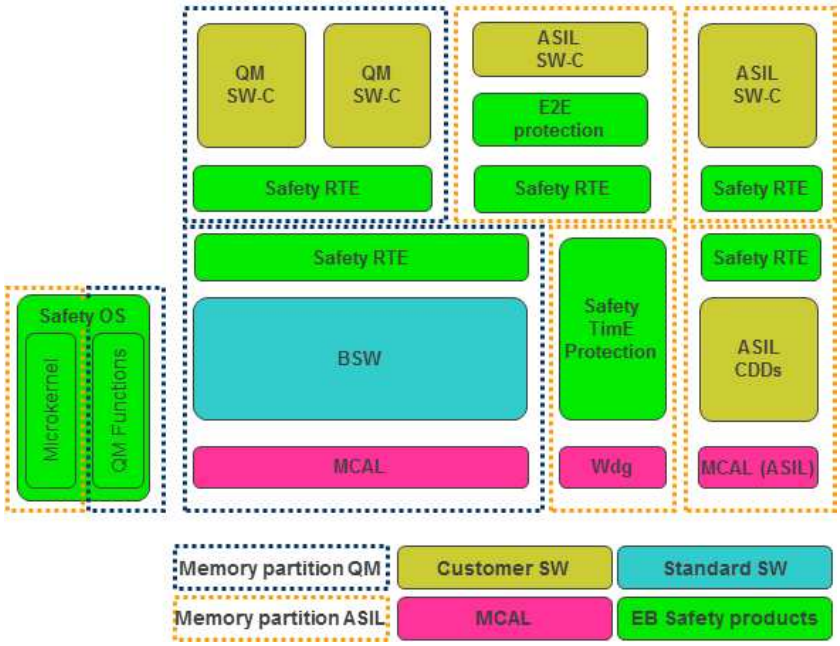

Figure 2: Safety architecture for an AUTOSAR ECU

to faults can provide spatial independence.

Temporal independence is based on mechanisms such as time monitoring, e.g. budget and deadline monitoring, or execution monitoring, e.g. alive supervision and control flow monitoring. These safety mechanisms are implemented in the operating systems or as a safety mechanism which is also independent and separated from the operating system.

2.3 Message Protection

Protection of messages is defined also in chapter "D.2.4 Exchange of Information" of ISO 26262 [ISO11b]. Failure scenarios for safety-critical ECUs include corruption, repetition, loss, delay of information.

The most commonly implemented approach is an end-to-end protection of the messages and data transmitted within an ECU or between ECUs. In an AUTOSAR context this enables the application to stay independent of the communication mechanisms and communication busses.

An implementation of the AUTOSAR end-to-end protection library including some extensions, which is developed according to SIL-3 and ASIL-D, provides the commonly used safety mechanisms for message protection.

3 Security Mechanisms

Security-critical systems have different needs than safety-critical systems as the "opponent" changes. The intention of a safety-critical system is to prevent harm emanating from the system itself whereas a security-critical systems intent to prevent harm to the system from outside, e.g. from a malicious human attacker.

In ideal world the "threat model" of a safety-critical system can be determined up-front during system design, analysis and implementation. The risk emanating from a safety-critical system is usually based on systematic or random hardware faults or systematic faults in software. Such a threat model is iteratively enhanced to include previously unknown faults, but basically one could call it "static".

Security-critical systems however need to deal with "dynamic" threat models as the methods used by an attacker are not known during system creation and can significantly improve during system operation. The major factors are long-term use of automotive systems, evolution in computing power, but most importantly the ingenuity of the attackers. Usually this makes security-critical systems more complicated to build and analyze than safety-critical systems.

However, the responsibility of the basic software doesn't change: provide mechanisms for ensuring the integrity of the system. This can be achieved by extending software architectural mechanisms established for safety-critical systems as well as adding new mechanisms especially designed for security topics. Most importantly, synergies can arise from the two analyses, as both analysis methods are risk-based. The detection of security breaches can also uncover possible safety issues and vice-versa. An example are typical stack overflow attacks, as discussed in more detail below. This chapter extends basic software architectures designed for safety-critical systems to security-critical systems.

3.1 Memory Read and Execution Protection

A good example for combined safety and security threats are stack overflows. They either originate from a fault in the software, which would be a safety issue, or they can be provoked by an external attacker. In both cases, such a scenario is part of the analysis and needs to be prevented by design or at least be detected during run-time.

Our architecture proposal shows solutions for many common safety and security topics. Staying with the stack overflow example, the EB tresos Safety OS prevents stack overflows using memory protection. While this still gives rise to possible denial-of-service attacks, it is not possible to bring the system into an undefined state. The overflow is detected and prevented, and the microkernel of the Safety OS invokes a pre-defined error handling ensuring that the ECU is transferred into a well-defined safe state or a degraded mode. The freedom-from-interference argument that is commonly used in safety-related mixed-criticality systems can thereby be used to ensure the separation of the critical safety mechanism from the vulnerable software part that is exposed to the "outside world".

Besides the MPU's write protection mechanisms, which is the basic use case for safety-critical software, modern CPUs also allow to activate read protection and even execution protection. The former can be used to ensure that security-critical data is only accessible by those software components that actually need it. This data can e.g. contain encryption keys or authentication data that must be protected against the software parts that have access to the network. A recent example for such an exploit is the "Heartbleed" vulnerability of OpenSSL, where the attacker can read sensitive authentication data via a maintenance function that is entirely unrelated to OpenSSL's encryption functionality.

The execution protection provides a further barrier against the attacker: the code dealing with the security-related data can be marked as non-executable, unless it is actually scheduled to run. In addition, adding execution protection to the stack also closes a commonly used door for intrusions into systems. Both mechanisms together thus eliminate two possible attack vectors: First, making unrelated code read and expose data that it is not considered to be read. Second, trying to execute the code involuntarily from a different context is actively prohibited.

3.2 Message Authentication

As already mentioned in chapter 2.3 in the safety context, a major issue is the reliability of message data. Since the scope of "opponents" is broadened in the security context, one has to assume an attacker has control over all communication channels in the sense of an attacker model according to Dolev-Yao (see [DY29]). This assumes particularly that an attacker can:

- Obtain any message passing through the network and read, modify or delete it
- Impersonate a legitimate user and thus is able to initiate a communication with any other user
- And therefore replay or delay messages

In the following we use the definitions for data integrity, authentication, entity authentication, data origin and message and transaction authentication given in [MVO96].

3.2.1 Authenticity and Integrity

Derived from the attacker model and according to [AUT14b], chapter 3.2 and 4.2 the major goals from a security perspective for message protection can be summarized as authenticity of entities and messages, data integrity, data freshness and message confidentiality (see chapter 3.3).

Data origin authentication or message authentication techniques provide assurance to a receiving party of the identity of the party which originated the message. Data origin authentication implicitly provides data integrity, since, if the message was modified during

transmission, communication partner ‘A’ would no longer be the originator (see [MVO96], chapter 1.7).

The most commonly used technique which is also proposed as the default method in [AUT14b] to achieve message authentication and integrity is a Message Authentication Code (MAC) and more specific a CMAC [Dwo05] based on AES [UDoCN01] with an adequate key length. As shown in Figure 3 the message on the bus is concatenated with (a part) of the MAC on sender side and can be verified on the receiver side.

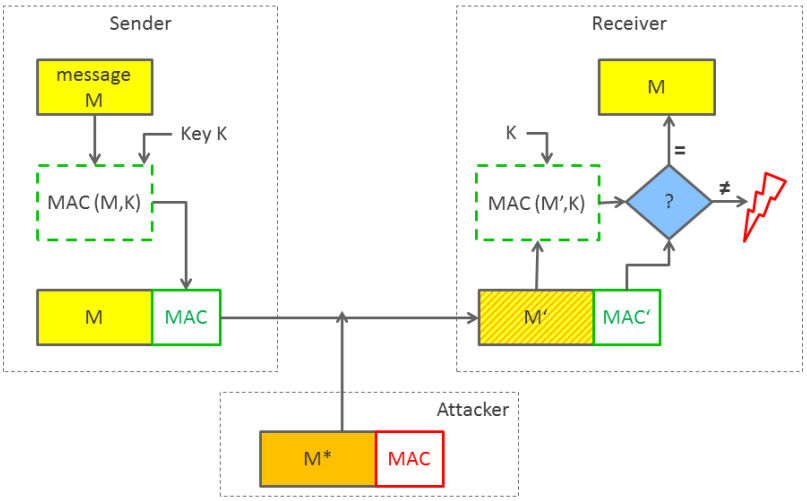

Figure 3: Message authentication code

3.2.2 Freshness

Using MACs alone does not protect from messages being recorded at a certain point in time and played (delay) or replayed by an attacker later. To prevent that an attacker simply records an authentic message of integrity and replays it at any point in time, each proof of authenticity has to be unique. This is achieved by adding a so called freshness value into the MAC generation process on sender side which allows the receiver to detect replayed messages on his side. Different sources can be used as freshness value, e.g. a monotonic counter or a reliable timestamp value (see [AUT14b]).

An implementation of the AUTOSAR concept ‘Secure Onboard Communication’ (see [AUT14b]) provides the commonly used security mechanisms for message protection ensuring authenticity, integrity and freshness of messages.

3.3 Message Privacy

At the moment, the AUTOSAR Secure Onboard Communication concept does not contain default use cases requiring message privacy. However increasing connectivity of vehicles and at the same time increasing integration of personal devices like smartphones into the vehicle environment will lead to a growing need for solutions providing privacy for various data.

Message privacy can be obtained by using, either symmetric or asymmetric (public key cryptography) encryption methods.
The most well-known and thoroughly analyzed symmetrical encryption method for current use cases is the Advanced Encryption Standard (AES) [UDoCN01]. State-of-the-art solutions for ECUs with higher security requirements use cryptographic functionality implemented in hardware, e.g. Secure Hardware Module (SHE) or more general Hardware Security Modules (HSM). Due to the standardized layered architecture such hardware peripherals can easily be made available in an AUTOSAR stack.

For asymmetrical cryptography, there are standards like RSA and elliptic curve cryptography [MVO96]. Compared to symmetric algorithms they are more computationally intensive and no standardized cryptographic hardware peripheral for automotive domain is currently available.

4 Extended Safety and Security Architecture

As seen in chapters 2 and 3 message protection plays an essential role in safety as well as security related applications. For both aspects protection against corruption, repetition, loss and delay of information must be ensured.

However the goals of both protection methods differ. For safety aspects the path from the originating to the receiving software application shall be protected against transmission and processing errors both on bus level as well and particularly as throughout the basic software stack on sender and receiver side. Message protection mechanisms with regard to security aim to protect messages on the bus against manipulation attacks. Protection within the software stack itself is not a goal of currently proposed security mechanisms (see [AUT14b]).

Therefore different methods are used to address safety and security aspects independently from each other. For protection against safety issues a checksum based end-to-end (E2E) protection mechanism specified by AUTOSAR (see [AUT13]) is most commonly used.

Protection against malicious attacks is done as specified in [AUT14b] and explained in sections 3.2.2 and 3.2.1 by appending and transmitting (parts of) a MAC value to the actual message data.

If a message is both safety and security relevant at the same time, currently two individual and independent attributes of significant length have to be added to the actual message data.

4.1 Safety and Security Architecture for Message Protection

To optimize the usage of available and already scarce bandwidth we propose the following two architectures for combined message protection with regard to safety and security. Since a MAC incorporating a freshness value can cover all aspects provided by a CRC value and extend these attributes by security relevant extensions as described in chapter 3 a MAC is used to protect messages on bus level.

The remaining challenge is to close the protection gap for safety protection between the verification of the MAC and the sending/receiving application.

4.1.1 Autosar E2E and SecOC modules

The safety protection gap arises because Autosar Secure Onboard Communication (SecOC) module is located on PduR level. To protect the path from PduR to the receiving application (usually a SWC above the RTE) we use the already established E2E mechanisms as shown in figure 4. On the bus only the security extension using a MAC is present.

Therefore SecOC does not simply cut off the MAC after successful verification - the default behaviour of SecOC – but converts it to a safety CRC. The application can then verify the faultless reception of the message in the same way as if the message is just safety relevant and protected by E2E mechanisms on the complete path.

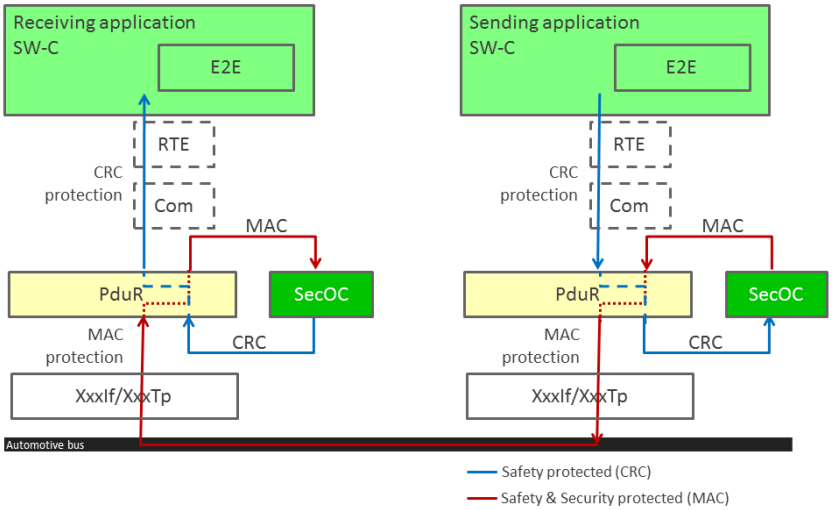

Figure 4: Architecture combining E2E and SecOC modules to efficiently protect messages

This architecture is compatible to both safety and security concepts in AUTOSAR enabling the use of all scenarios and features of the original solutions. It extends the SecOC just to transform a MAC into a CRC value and vice-versa.

4.1.2 End-to-end protection with security methods

An alternative setup EB specialists use to realize end-to-end protection with safety and security allocation is shown in figure 5. Here the safety module is replaced by the security module. To protect a message also through the basic software stack the latter is placed close to the application. AUTOSAR 4.2.1 plans to integrate E2E as part of the serializer/transformer chain ([WA14]). Figure 5 shows both architecturally similar approaches for such a scenario, with the new version for AUTOSAR 4.2.1 on the right-hand side.

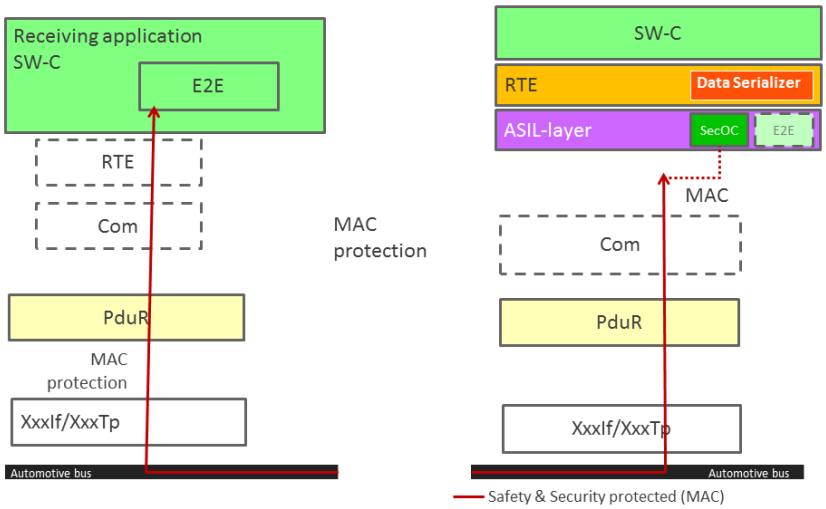

Figure 5: Architecture replacing E2E safety with security mechanisms

This approach, while simpler on paper, requires to change the integration of several adjacent but crucial disciplines like key management, communication behaviour etc. compared to the original standardized concepts.

4.2 Extended Safety and Security Architecture

An architecture combining message protection with the memory read and execution protection mechanisms described in chapter 3.1 is shown in figure 6.

In our implemented application the message protection variant from chapter 4.1.1 was combined with memory read and execution protection provided by EB safety products. To calculate and verify the CMAC we used a SHE hardware peripheral which was integrated into the AUTOSAR stack via a so called CryShe module.

The approach allows to mix security and safety software (also different (A)SIL level) as well as software with normal quality requirements (QM) and enables the integration of

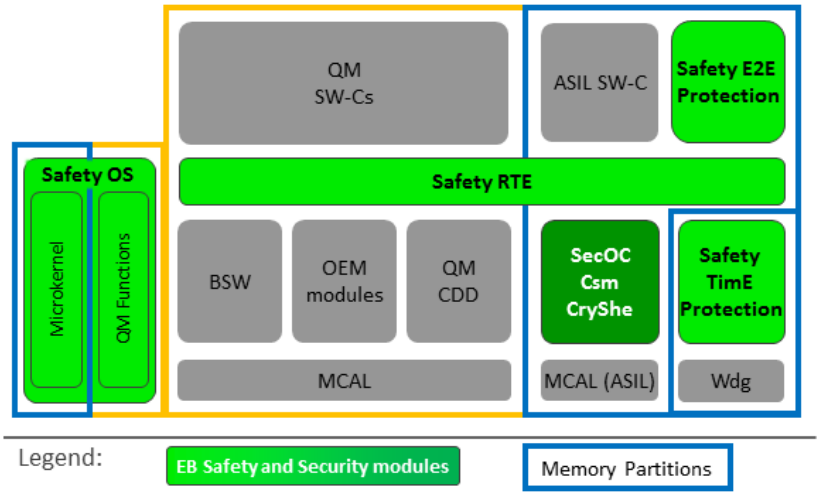

Figure 6: Architecture combining safety with security mechanisms for message, memory and execution protection

"black-box" software (e.g. from an OEM or another supplier) while re-using most building blocks of the standard architecture.

5 Systems and Software Engineering

For safety-critical systems techniques for performing safety analyses are well-established and these are based on approaches known from risk management combined with specific analysis techniques, e.g. for software architectures or the implementation itself. Risk analyses of the system and the implementation are the most important "add-on" in the development process of safety-critical systems compared to standard development processes.

Security analysis of systems is already established in other domains, e.g. in the IT sector. For many automotive ECUs a dedicated security analysis is often not part of the standard development processes yet. However both analysis types are similar in the sense that they are risk-based. These analyses become more alike the closer they are performed to the actual implementation: starting with the software design the formal inspections are nearly identical.

Both domains require rigid software engineering and many threats are already prevented using established processes and methods. For a good summary see a recent response to the heartbleed issue by David Wheeler [Whe14].

When engineering security-critical systems a very important aspect is to look at the complete chain allocating security requirements. The system in question is never only the

embedded target but affects all modules communicating with the functions in question as well as providing and storing crucial data, e.g. key management and key distribution systems, IT infrastructure, cloud services, etc. As a consequence the IT security in the companies needs to be linked with the devices.

A good methodological starting point for risk identification, analysis and risk treatment is the Information Security Management System (ISMS) standard – ISO 27001 ([iso14]).

6 Conclusion

In this paper we presented a practical approach to extend a software architecture designed for safety critical systems based on AUTOSAR to be used in security critical systems.

We have pointed out the mutual influence between safety and security aspects which are recognized already in different standardization groups as well as by regulatory bodies and political institutions. José Manuel Durão Barroso summarized them spot on by pointing out "[...] there is no safety without security and vice-versa", see [Jos12].

Already well-established mechanisms in safety development, e.g. in the definitions of "independence" in IEC 61508 [IEC10] or "freedom from interference" and "criteria for coexistence of elements" according to ISO 26262 can be reused. For applications in common security use cases they are extended by cryptographic methods to ensure e.g. entity and message authentication as well as advanced integrity checks.

We have shown, that one can achieve a significantly increased level of safety as well as security by combining already available solutions from both worlds and consequently applying rigid software engineering processes and methods derived from functional safety development and extending them to security use cases.

7 Outlook

We are currently extending the software architecture approach further with a secure hypervisor to support different operating systems to further support popular "apps" which can be used directly in a vehicle. This also includes merging different domains, e.g. AUTOSAR and Linux-based systems as shown in Figure 7 which can communicate in a controlled (secured) way via an inter OS communication module.

In contrast to a safety architecture the adaptation to new security threats will be a never ending process which needs constant monitoring and adaption to newly emerging threats. This implies the capability of the system to be upgradeable and maintainable during the whole operative life-cycle, which is interesting especially from a safety-critical point of view.

With the development of more and more connected features relying on data from various sources this trend will accelerate further and will increase the need for combined safety

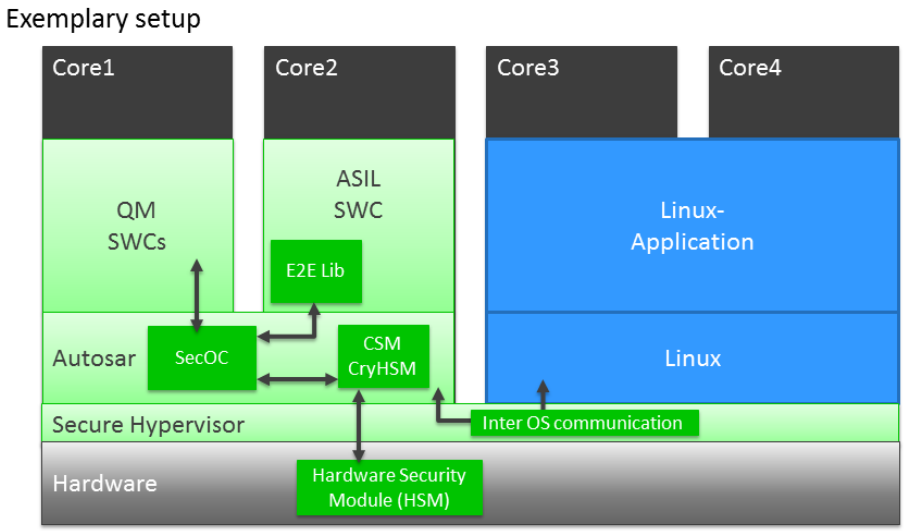

Figure 7: Possible future safety and security architecture supporting different operating systems

and security approaches. These need to be based on standardized software architectures that provide mechanisms for the “integrity” of the system.

Security related systems differ from today’s software architectures used in the automotive domain which are usually based on direct and synchronous API calls. However for memory read protection many API functions need to be converted to asynchronous calls. This change is also motivated by moving from single-core to multi-core processors. The resulting software architectures are challenging since a fundamental property of a safety related system is a limited complexity of the dynamic aspects of the software architecture, e.g. a clearly defined or even static schedule.

References

[AUT13] AUTOSAR. AUTOSAR specification of SW-C End-toEnd Communication Protection Library, 2013.

[AUT14a] AUTOSAR. AUTOSAR - AUTomotive Open System ARchitecture, 2014.

[AUT14b] AUTOSAR. AUTOSAR concept - Secure Onboard Communication, 2014.

[Bun09] Bundesgerichtshof. BGH, Urteil vom 16. Juni 2009, Az.: VI ZR 107/08, 2009.

[Dan10] Christian Daniel. Angreifbare Mobilität: Wie ”online” ist ein modernes Auto und welche Sicherheitsrisiken ergeben sich?, 2010.

[Dwo05] Morris J. Dworkin. SP 800-38B. Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. Technical report, Gaithersburg, MD, United States, 2005.

[DY29] Danny Dolev and Andrew C. Yao. On the security of public key protocols. 29, 1983-03-29.

[FDA13] FDA. FDA security alert, Cybersecurity for Medical Devices and Hospital Networks, 2013.

[GM18] M. Galla, Thomas and Alexander Much. Targeting Freedom from Runtime Errors for AUTOSAR-Based ECU Software. Mathworks Automotive Conference, 2012-04-18.

[HJMA14] David Haworth, Tobias Jordan, Alexander Mattausch, and Much Alexander. Freedom from Interference for AUTOSAR-based ECUS: a partitioned AUTOSAR stack. Automotive Safety and Security, 2012-11-14.

[IC13] ICS-CERT. ICS-CERT ICS-ALERT-13-164-01 - Medical Devices Hard-Coded Passwords, 2013.

[IEC10] *Functional safety of electrical/electronic/programmable electronic safety-related systems: part 6 : guidelines on the application of IEC 61508-2 and IEC 61508-3.; 2.0*. IEC, Geneva, 2010.

[ISO11a] ISO. ISO 15026, Systems and Software Engineering – Systems and Software Assurance, 2011.

[ISO11b] ISO. Road vehicles – Functional safety, 2011.

[iso14] ISO/IEC 27001:2013/Cor 1:2014: Information Security Management System (ISMS) standard. *Online: http://www.iso.org/iso/iso27001*, 2014.

[Jos12] EU Action on Nuclear Safety, European Commission - SPEECH/12/227 27/03/2012, 2012.

[KCR+10] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon Mccoy, Brian Kantor, Danny Anderson, Hovav Shacham, and Stefan Savage. Experimental Security Analysis of a Modern Automobile. Seattle, Washington 98195–2350, 2010. In Proceedings of IEEE Symposium on Security and Privacy.

[MA09] Alexander Mattausch and Much Alexander. Standardized Software Architectures for Safety ECUs. Forum Funktionale Sicherheit, 2013-10-09.

[MV13] Charlie Miller and Chris Valasek. Adventures in Automotive Networks and Control Units. *Last Accessed from h ttp://illmatics. com/car_ hacking. pdf on*, 13, 2013.

[MVO96] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., 1996.

[RTC11] RTCA. DO-178C – Software Considerations in Airborne Systems and Equipment Certification, 2011.

[UDoCN01] National Institute of Standards U.S. Department of Commerce, Information Technology Laboratory (ITL) and Technology (NIST). FIPS Pub 197: Advanced Encryption Standard (AES). Technical report, Gaithersburg, MD, USA, 2001.

[WA14] AUTOSAR WP-A3. AUTOSAR concept - E2E extension, 2014.

[Whe14] David A. Wheeler. How to Prevent the next Heartbleed, 2014.