RT&A, No 4 (65)
Volume 16, December 2021

SAFETY vs. SECURITY – WHY ARCHITECTURE MAKES THE DIFFERENCE

Jens Braband

•

Siemens Mobility GmbH
Jens.braband@siemens.com

Hendrik Schäbe

•

TÜV Rheinland InterTraffic GmbH
Schaebe@de.tuv.com

Abstract

Cybersecurity plays an increasing role. This also holds true for safety systems. Hence, it is necessary to combine systems that fulfill security and safety requirements. These requirements are partially contradictory. Safety related software will not be changed in an ideal world, whereas security software needs almost permanent updates. This leads to problems that are hard to solve. Different approaches have been proposed by different authors. In this paper we will show, how a suitable architecture can be applied to satisfy the security as well as the safety requirements. We consider some examples of such architectures and show, how systems can be constructed that on the one hand side contain a "golden" code for safety that is not changed and on the other hand side security software that can easily be patched, not touching the "golden" code.

Keywords: Safety architecture, cybersecurity architecture, patching

I. Introduction

It is often claimed that safety and security shall be separated as much as possible but coordinated well. This is also the mantra of new TS 50701 [1]. Also, this is part of the requirements of the EN 50129 [2], see section 7.2 and table E.4 entry 1, which require the separation of safety and non-safety parts of the system. Since usually a security component is not a safety component in a first approximation, this is just what the standard requires.

But the problem is, how to achieve a good implementation of safety and security requirements. There are concepts, that that proper management is the optimal solution, others believe in coordinated lifecycles, and there are also other approaches [3,4,5,6].

However, in the view of the authors, architecture is the decisive factor. Safety and security architecture makes the difference. It is hard to generalize the good practices that are known, but this paper tries to give a few patterns.

II. The "Detect Single Faults" Pattern

Let us consider the first example, which is more or less a straightforward solution. We start with a well-known qualitative design pattern, which works for many safety-related systems. This is the often so-called "fail-safe" system, see EN 50129 [2 for the requirements. See also [7] for further research. This pattern has also its merits for safety vs. security.
Assume, one adds a single component K to a class 2 system S (or zone) according to EN 50159 [8], see figure 1.

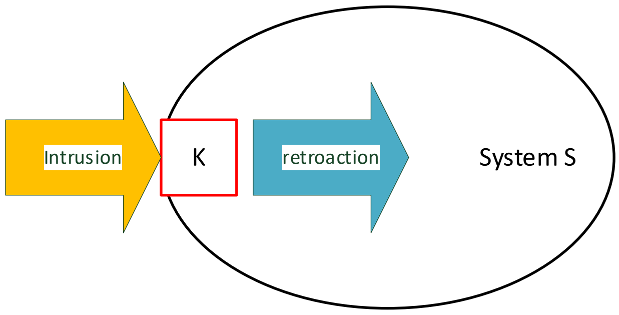

Figure 1: *Adding a security component K to a safety system S*

Even if one just adds K to S, some safety homework still needs to be done, as K may have impact on S even if K does not implement a function for S. For example K may increase latency, decrease reliability etc. The situation gets worse when K is connected to some outside network. Then, in addition a security risk assessment must be carried out, as intrusion may now be possible. Let us assume that this has been managed. However, the solution is not yet complete. One has still to make sure that the security functions implemented in K are functioning according to their specification and that they persist doing this.

Normally the safety standards require that the security mechanisms of K are monitored by S. But the detailed requirements depend on the function that K implements and its architecture.

Example 1: Single-Component-Architecture. Assume K is a filter or firewall just as a kind of gatekeeper that lets only permitted traffic pass (simple whitelisting). If there is a reasonable single failure mode that (partially) deactivates the function, then it is very likely that some monitoring has to be included and that results safely need to be checked – possibly this is not done by a technical function)

Example 2: Two-Component-Architecture: Assume K encrypts transparently all traffic from S to a neighboring zone S*, which has a counterpart K*. If now K fails to decrypt or encrypt any messages, then this will be immediately noticed at the other network when messages start missing. So given a sufficient traffic flow, it is highly unlikely that both components suffer from similar faults with the same effect within a few milliseconds. One can neglect this risk and there is no need to implement any additional monitoring on the safety level. However, it is necessary to ensure that there are no common causes in the two components or in supporting processes as e. g. maintenance that led to the same failure on both sides. Examples are the deactivation of encryption on both sides, use of default keys outdated algorithms etc.

III. The "Safety Channel" Pattern

In this section, we consider another possible architecture. Patching is a particular hot topic when considering systems that need to fulfil safety and security requirements as well. In safety applications, everybody is reluctant to change the certified "golden code", while in security some applications shall be updated or patched every day. This seems to be a contradiction. It can be solved using an appropriate architecture.

Example 3: Assume one has a safety application, which needs to be protected by a virus checker (VC). This situation is equivalent to a situation, where 3rd party SW needs to be run on the same entity, be it a computer, a kernel or a virtual machine. Assume you need a majority of votes of the different entities for a safety critical decision, e.g., moving a switch.

The basic idea is to split the population of entities into two tribes: the entities labeled N are never changed - or only when the safety application needs to be updated, the entities labeled P can be patched as often as necessary. Of course, some integration tests need to be carried out before patching.. Additionally, the architecture contains voters V that check the outputs before execution. It goes without saying, that all components must be type checked before first operation.

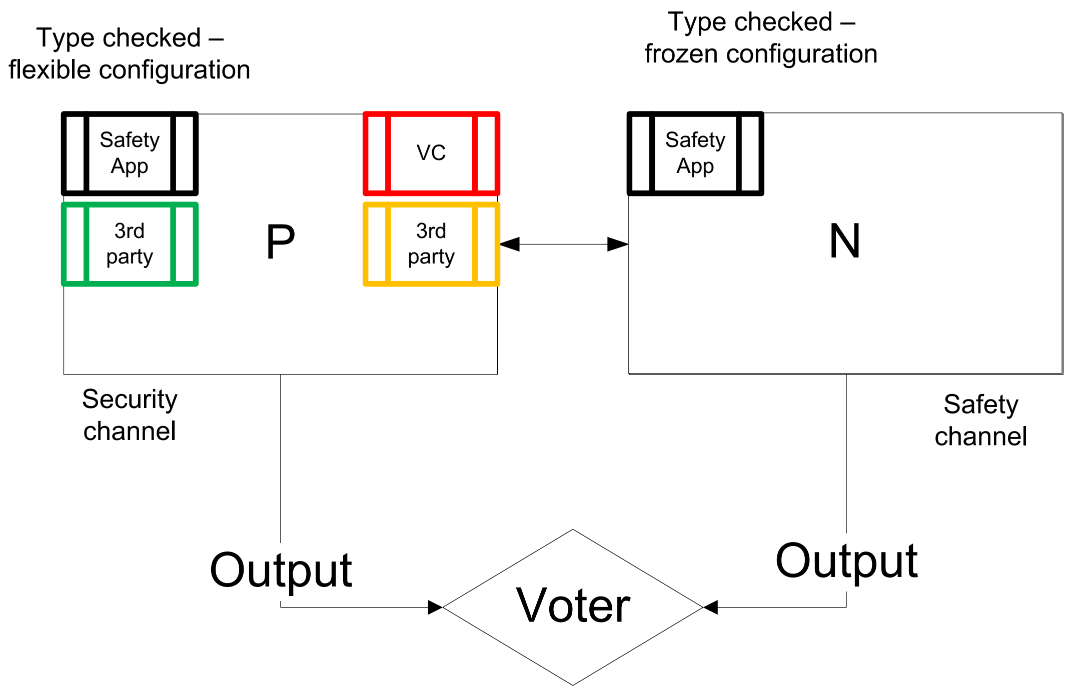

Figure 2: *Patch-friendly architecture*

Figure 2 shows this architecture for a 2002 configuration with one N and one P entity. Both have the same safety application, but on entity P has another software is implemented that needs regular updates. Entity N is sealed, physically and logically protected, e. g. tamper-proof. It is never touched (unless you want to change the safety app). In P the safety application is also never changed. On both channels this may be checked, e.g., by using a hash code on the application or other means as a command from the voter requiring a specific response from both channels, which must coincide. Functional differences in both channels can be detected by the voter command. Now whenever a safety decision needs to be taken, both N and P must agree, which is checked by the voter. So, a final decision of the system is only possible, if the unchanged safety application of N agrees with the decision of the application in P. And thus, it does not matter what other SW runs on P or if it is patched or not. If P was hacked or tampered with or the safety app influenced by the other apps, then P could not change the decision of N. But also, N can't take any decisions of its own, it always needs an agreement of P, the channel that is virus protected etc.

This may now be generalized e. g. to 2 N and 2 P channels demanding that always a majority agrees etc.

Note that some architecture elements have been left out for the clarity of the argument e. g. the inputs, power supply, communication as well as separation between the channels. So, there is still some homework to do. Nevertheless, this architecture is suitable to roll out patches with this architecture.

IV. The "Mixed Architecture" or "EN 50159" pattern

In some cases, it is not possible or not wanted do strictly divide the safety and the security components.

Assume that in the safety part some security functions are integrated, i.e. because on the application level some encryption or message authentication is running. This type of functions is implemented since measures described in EN 50159 [8] are implemented. One must note that EN 50159 is not a cybersecurity standard, it is for safety related communications. So, the measures, although partially the same as in cybersecurity, are dedicated against technical processes that might influence or degrade communication. The mechanisms are others than with hackers, see Braband and Schäbe [9]. Nevertheless, we arrive at the following architecture.

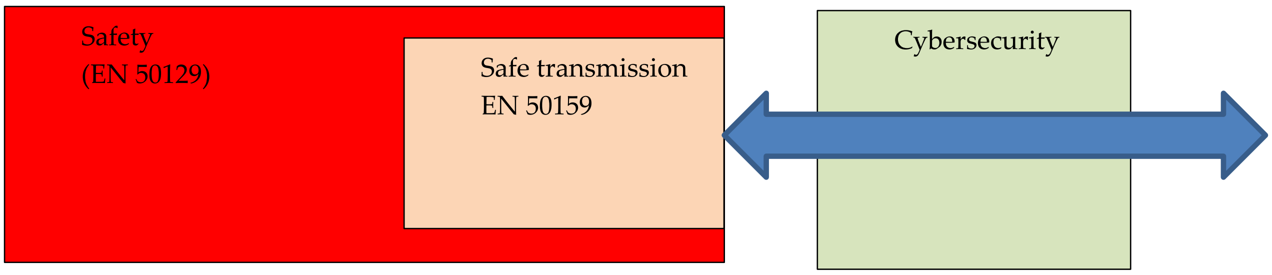

Figure 3: *Mixed Architecture*

Assume that such a system is connected to a similar one. This means, that the safe transmission part of the system as well as the safety parts should not be changed (golden code). All patches should be applied in the cybersecurity part, which then also must compensate for those measures for cybersecurity that cannot and will not be implemented in the safety block.

Example 4: Let us consider the following example. In the safe transmission block, there is an encryption algorithm to protect the data with regards to confidentiality and authenticity (like ETCS). Of course, this protection would not be complete since a hacker might attack the safety part of the system and get access to the data and the code. Therefore, the additional security module is necessary to ensure complete cybersecurity protection. In this example, the security part is split: a never change tribe consisting of the safety part – including the safe transmission part and a patch tribe consisting of the security part. This is a bit similar to example 2 described above. In order to cope with the increasing possibilities of hackers, the security parts is patched. This might even lead to a situation, where the encryption algorithms implemented in the safe transmission part will become superfluous, since in the security part an additional encryption method is installed. The system might thus evolve to the situation described in example 1. However, it is also possible, when carrying out an update of the safety part, also to update the safe transmission part by implementing a more efficient encryption algorithm. This makes it possible, to remove it then from the cybersecurity part.

From example 4, we see that it is only partially possible to design combined systems. Only methods that would not require permanent patching can be implemented in the safety part. Regarding the algorithms, they must work with a certain reserve, i.e., not only provide the simplest and most basic solutions to the problems. In the safety part, encryption can be used, where a method needs to be chosen that cannot be broken within the next months. Here, one must also not only look for the time that the method can withstand brute force attacks, but backdoors and exploits need to be absent. But of course, these basis security measures in the safety part do not need to be perfect. The main security protection is implemented in the cybersecurity part. Other components as interface drivers, firewalls etc. should not be contained in the safety part since they are candidates for permanent patching.

The architecture discussed in [10] can also be seen as an example of this architecture. In [10] the authors proposed an architecture, where a complete separation of safety and security issues had been carried out. The security mechanisms generate conduits through which the safety functions could be implemented. This leads to the onion skin model shown on figure 4. The onion skin model can only be implemented in this rigorous manner, if a struct separation of safety functions and security functions is possible. In that case, it is the most efficient architectural solution.

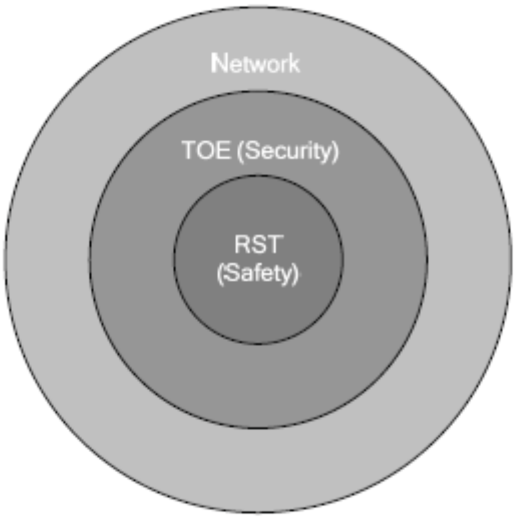

Figure 4: *The onion skin model, taken from [10]*

V. Conclusion

In this paper, we have presented examples system architectures that allow to fulfill the requirements arising from security as well as from safety. We have shown, that with the help of an appropriate architecture, the dilemma of conflicting requirements can be solved in an efficient manner. Surely, not each architecture is applicable for each situation. Therefore, we have presented several examples. We believe that by this approach the problem can be solved in an efficient manner.

References

[1] CENELEC: Railway applications – Cybersecurity, TS 50701, 2021.

[2] EN 50129 Railway applications - Communication, signalling and processing systems - Safety related electronic systems for signaling, 2018.

[3] Glas, B., Gebauer, C., Hänger, J., Heyl, A., Klarmann, J., Kriso, S., Vembar, P. and Wörz, P., (2015). Automotive safety and security integration challenges. In: Klenk, H., Keller, H. B., Plödereder, E. & Dencker, P. (Hrsg.), *Automotive - Safety & Security* 2014. Bonn: Gesellschaft für Informatik e.V.. (S. 13-28).

[4] Kharchenko V., Dotsenko, S.,Illiashenko O., Kamenskiy S., (2019), Integrated Cyber Safety & Security Management System: Industry 4.0 Issue, *10th Internaional Conference on Dependable Systems, Services and Technology (DESSERT)*, 5.-7. 6.2019, DOI: 10, 1109/DESSERT.2019.8770010

[5] Novak, T. and Treytl, A. (2008), Functional Safety and System Security in Automation Systems A Life Cycle Model, *Proceedings of ETFA - 13th IEEE Conference on Emerging Technologies & Factory Automation*, 311-318, DOI: 10.1109/ETFA.2008.4638412

[6] Novak, T., Treytl, A. and Palensky, P., (2007) Common Approach to Functional Safety and System Security in Building Automation and Control Systems, *Proceedings of ETFA. IEEE Conference on Emerging Technologies and Factory Automation 2007*, 1141-1148, (C) IEEE 2007, DOI 10.1109/EFTA.2007.4416910

[7] Gayen, J.T. and Schäbe, H. (2008). (Mis-) conceptions of safety principles, *ESREL Proceedings Safety, Reliability and Risk analysis*, 2: 1283-1291.

[8] CENELEC: Railway applications Communication, signalling and processing systems Safety-related communication in transmission systems, EN 50159, 2010.

[9] Braband, J. and Schäbe, H. (2016). Probability and security – pitfalls and chances, *Safety and reliability*, 36:3-12

[10] Bock, H.-H., Braband, J., Milius, B. and Schäbe, H. (2012). Towards an IT Security Protection Profile for Safety-Related Communication in Railway Automation, *Lecture Notes in Computer Science, Computer Safety, Reliability, and Security*, 7612: 137-148