



Understanding Common Automotive Security Issues and Their Implications


Downloaded from: <https://research.chalmers.se>, 2025-06-18 04:02 UTC

Citation for the original published paper (version of record):

Lautenbach, A., Almgren, M., Olovsson, T. (2019). Understanding Common Automotive Security Issues and Their Implications. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 11552 LNCS: 19-34.
http://dx.doi.org/10.1007/978-3-030-16874-2_2

N.B. When citing this work, cite the original published paper.

Understanding Common Automotive Security Issues and Their Implications

Aljoscha Lautenbach(^[0000-0001-5666-9940], Magnus Almgren^[0000-0002-3383-9617], and Tomas Olovsson^[0000-0001-9548-819X]

Chalmers University of Technology, Gothenburg, Sweden
{aljoscha, magnus.almgren, tomas.olvsson}@chalmers.se

Abstract This is the authors' version of this paper. The final authenticated version is available online at https://www.doi.org/10.1007/978-3-030-16874-2_2.

With increased connectivity of safety-critical systems such as vehicles and industrial control systems, the importance of secure software rises in lock-step. Even systems that are traditionally considered to be non safety-critical can become safety-critical if they are willfully manipulated. In this paper, we identify 8 important security issues of automotive software based on a conceptually simple yet interesting example. The issues encompass problems from the design phase, including requirements engineering, to the choice of concrete parameters for an API. We then investigate how these issues are perceived by automotive security experts through a survey.

The survey results indicate that the identified issues are indeed problematic in real industry use-cases. Based on the collected data, we draw conclusions which problems deserve further attention and how the problems can be addressed. In particular, we find that key distribution is a major issue. Finally, many of the identified issues can be addressed by improved documentation and access to security experts.

Keywords: Automotive Application Development · Automotive Security · Expert Survey

1 Introduction

Imagine that, while driving on the highway, the driver seat suddenly starts to slide back and forth, and the seat adjustment controls are not responding. Clearly, the driver will have problems to drive the car safely: she may be unable to reach the brakes in a critical moment, or her movement may be so restricted that it is impossible to steer correctly. Perhaps the driver would be able to handle the situation for a short time, but after a while she would become fatigued from having to constantly adjust her body to the changing seat position, which can lead to dangerous situations. There are many safety-critical systems in a car, but as this example shows, even not directly safety-critical systems such as the seat adjustment system can have a negative impact on safety when attacked.

In the last few years, a growing number of cyber security problems have been discovered in automotive systems. The first systematic security investigations started early this millennium [24], but only the more recent works by Checkoway et al. [10,4], Miller and Valasek [13] and others brought the problem to the attention of a wider audience. Automotive systems face the same challenge as all embedded systems in the wake of ubiquitous connectivity: their technology was designed when connectivity was limited, and malicious attackers were not seen as a serious threat since local access was required to do any damage [9]. Therefore, security mechanisms are missing and must now be added in retrospect [24]. A first step to help remedy this situation is the “SAE J3061 Cybersecurity Guidebook for Cyber-Physical Vehicle Systems” [17].

In general, automotive software engineers are well trained in the safety aspects of their work, but few have security training. This is complicated by the fact that, depending on the context, the same terms may have different meanings across disciplines [3,6,8,12,16]. Without security training, even cyclic redundancy checks (CRCs) can easily be misconstrued as sufficient for security [25]. More generally, there is a broad consensus among security experts that implementing secure systems and using cryptographic libraries correctly requires finesse and training [2,11]. A recent study by Acar et al. [1] showed that the API of cryptographic libraries has a significant impact on the security of the resulting application code.

In this paper, we have used a conceptually simple example of a seat control application and enumerate issues that may arise or questions that need to be answered in the course of the development. This resulted in eight issues a developer faces, ranging from the open-ended act of understanding the threats to the much more concrete choice of key length for encrypted or authenticated data. These eight issues form the basis for a survey sent to automotive security experts, and the data is further analyzed before we present our recommendations and conclusions.

2 Methodology

Considering the design of a simple automotive control application, such as a seat control application, there are several security issues which arise naturally during the development process. We chose this simple use-case to capture the most common issues, and more elaborate use-cases will almost certainly unearth additional issues. In order to investigate to what extent the identified security issues are perceived to be problematic, we surveyed automotive security experts. We also questioned the experts to what extent some given recommendations address the issues. The survey was distributed to several contacts at seven automotive original equipment manufacturers (OEMs) and a small number of consultancy companies. Participation was voluntary and anonymous, and company affiliations were not tracked to preserve anonymity. This implies that participant distribution among the companies is unknown. The survey was roughly

structured into four main parts: (a) background questions, (b) introduction of the contexts, (c) questions for each issue and (d) questions on the recommendations.

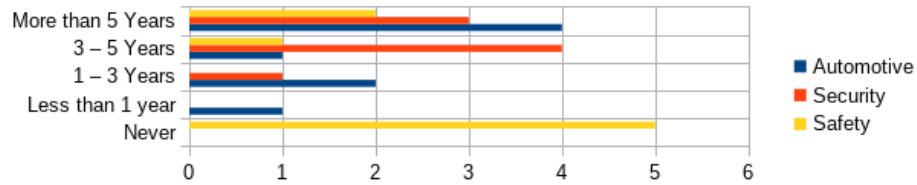
In part (a), we asked some demographic questions about the participants' background. This included questions on their number of years of experience with automotive systems, security, and safety respectively, and what their primary work task is (requirements engineering, software development, etc).

In part (b), we explicitly introduced three contexts, since some of the questions about the security issues can be understood slightly differently depending on each participant's interpretation of the context. The presented contexts were: (1) an independent context (i.e., "in a general sense"), (2) in the context of the participants' own work and (3) in the context of a scenario we called "simple automotive control application (SACA)". The scenario was described as follows.

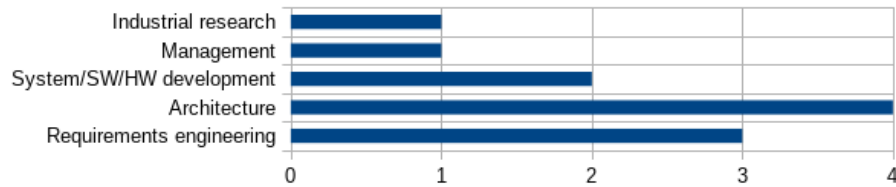
You are developing a simple automotive control application (SACA) that operates on two electronic control units (ECUs), which are connected via a CAN bus (max bandwidth of 1 Mbit/s). The two ECUs exchange messages periodically.

In part (c), the questions for each issue were presented. At this point it is important to point out that the issues were phrased in a neutral way as *activities* to avoid leading the subjects. For the same reason, illustrating examples and elaborations were avoided, with the assumptions that security experts would be familiar with the various difficulties that accompany each of the chosen issues. The issues were roughly grouped into the categories of "Design and architecture", "Programming" and "Parameters". The questions for each issue followed a fixed pattern. There were four questions per issue that can be paraphrased as follows:

1. Importance: How important is "activity X"?



(a) Participants' experience with automotive systems, security and safety



(b) Participants' primary work tasks (multiple choice)

Figure 1: Survey participants' background

2. Difficulty: To what extent do you agree that “X” is a difficult activity?
3. Frequency: How often do you perform “activity X” in your own work?
4. Comments: Do you have recommendations how to simplify/improve “activity or situation X”? Do you have any other comments? (optional free text)

The questions on importance and difficulty were asked for all three contexts, while the frequency question was only asked in the context of the person’s own work. The particularly interesting activities are those which are both important and difficult. An activity that is both frequent and difficult might also be of some interest, even if it is not perceived to be important.

3 Survey Participants

In total, eight industry professionals completed the survey.¹ Even though this is a rather low number of participants for a general survey, we argue that given the very specific target group of automotive security experts, eight is still a good number in this highly specific context. These are people with considerable expertise and knowledge of the area (see Fig. 1a). We considered to include non-experts in the survey, but felt that it would not benefit the quality of the results, and so it was deemed better to have relevant answers from a small but targeted group.

One demographic question was about the participants’ experience with automotive systems, security and safety, and the results are shown in Figure 1a, emphasizing that our target group has significant experience with security.

Figure 1b shows the distribution of the primary work tasks, which was a multiple choice question. Nobody answered that they primarily work with “Testing”, “Academic research” or “Other”, so these options are not shown in Figure 1b. It is notable that seven out of eight respondents work with either architecture or requirements engineering, which may introduce a certain bias. Furthermore, only two respondents indicated that they primarily work with system, hardware or software development. Much system development happens at suppliers, so this makes sense.

4 Common Automotive Security Issues

As outlined in Section 2, we identified eight common automotive security issues by considering the security needs of a simple networked control application, and we designed a survey with the aim to identify which of these issues may warrant deeper investigation.

For the *importance* and *difficulty* related questions of the survey, we will only present the answers for the person’s own context, the answers for the other two contexts are typically similar. In fact, there is a general trend that the importance of the issues is rated highest for the person’s own work, and importance in the

¹As stated earlier, we did not track company affiliations to preserve anonymity.

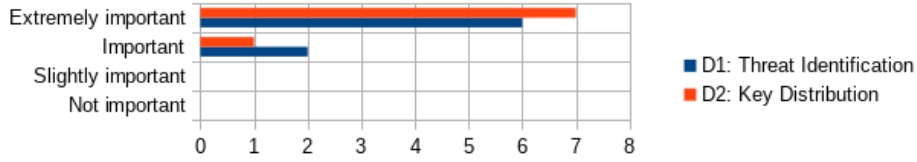
general context is rated lower than in the SACA scenario. The exact same trend can be observed for difficulty.

In the following subsections we present the eight identified issues together with the aggregated survey data, roughly grouped into *Design and architecture*, *Parameters* and *Programming*. Since the dataset is small, we present the survey answers per participant in the final subsection, and make some observations about possible correlations.

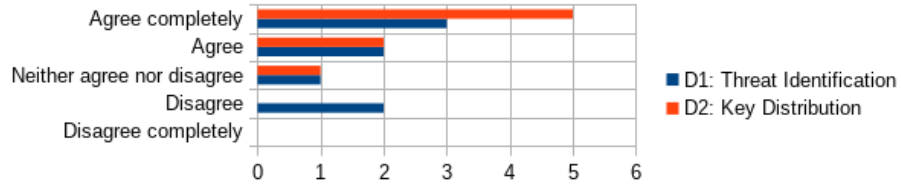
4.1 Design and Architecture

Designing secure systems involves several steps, the first of which is to identify the threats to the system. The next steps are to choose security measures to counter the identified threats and to implement those security measures. Without identifying the threats first it is difficult to choose appropriate countermeasures.

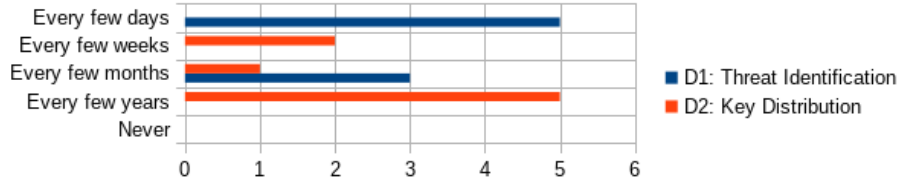
D1 Identification of Threats There are many different ways to gather the security requirements of an application, but they all require some experience and training. The identification of threats is typically paired with a risk assessment



(a) The number of answers to how important the investigated design issues are



(b) The number of answers to whether the investigated design issues are difficult



(c) The number of answers to how frequent the participants encounter the investigated design issues in their own work

Figure 2: Survey results for the design related issues

procedure, similar to the way hazard analysis is paired with risk assessment in safety engineering [7]. Therefore the identification of threats is the first issue to consider.

Once the threats have been identified and the security measures have been chosen, it is time to implement them. Many, if not most, security measures use cryptography in some form. For in-vehicle communication, the question of symmetric versus asymmetric cryptography is relatively simple to answer: symmetric cryptography is strongly preferred due to much better performance. Asymmetric cryptography still has a place in the vehicle for functions where speed is not critical, e.g., signatures for remote software updates, or when pre-shared keys are impractical, e.g., when communicating over the internet.

D2 Choice of a Key Distribution Mechanism When symmetric cryptography is used, the keys must be available at both communication ends before communication starts. There are several ways key distribution can be done. One possibility is to use an out-of-band channel (pre-shared keys). This is usually done during production. Alternatives are public key schemes such as the Diffie-Hellman key exchange protocol or certificates.

The choice of the key distribution mechanism has both practical and security implications. For instance, repair and maintenance is an important criteria to consider [15]. If an electronic control unit (ECU) breaks, maybe due to an accident, and keys on that ECU become inaccessible, there must be a way to handle this. Additionally, the automotive ecosystem includes many multi-tiered suppliers, and some ECUs that arrive at the vehicle manufacturer for assembly are essentially black boxes, developed to precise specifications. This raises the questions, who installs the pre-shared key, and how is it installed on other ECUs that need to communicate with the ECU?

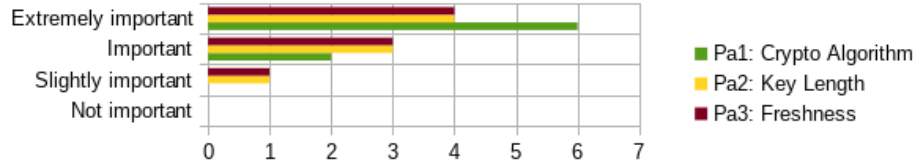
Survey Results Figures 2a to 2c depict the survey results for design and architecture issues. As Fig. 2a shows, there is strong agreement among the experts that these are important issues. There is also strong agreement that key distribution is a difficult problem to solve, while the opinions on threat identification are slightly split (Fig. 2b). The answers also indicate that threat identification happens frequently, whereas choosing a key distribution mechanism is rare. This somewhat reflects that many of the participants work with requirements engineering and architecture. The high frequency of threat identification may explain why some disagree that it is a difficult task.

4.2 Parameters

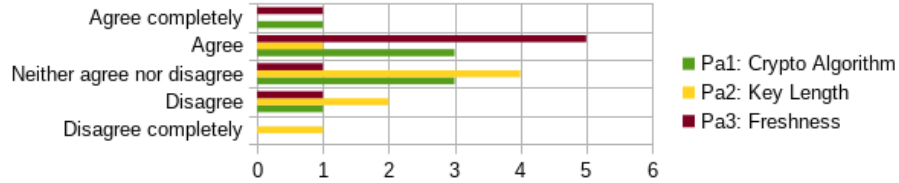
Once the basic design decisions have been made, several parameters have to be chosen to implement the chosen security mechanisms. Parameter choices can include the choice of a cryptographic algorithm, choosing an appropriate key length and choosing a mechanism to ensure *freshness*.

Pa1 Choice of Suitable Cryptographic Algorithms The choice of a cryptographic algorithm is not always straightforward. Cryptographers constantly try to find weaknesses in published algorithms, and an algorithm which was considered secure five years ago may not be so today, although this is often a gradual process. A good example for gradual deprecation is the SHA-1 cryptographic hash algorithm: first attacks have already been discovered in 2005 [23], and it has been considered weak for many years, but the first collision was only publicized in 2017 [19]. Since automotive products have a lifetime of 10 to 20 years, the algorithms must be chosen with care. Apart from pure security considerations, automotive systems have strict requirements for performance, and trade-offs must be considered.

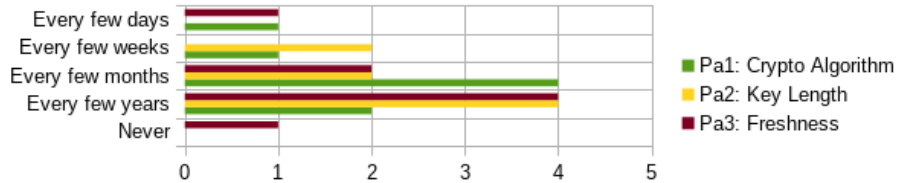
Pa2 Choosing a Suitable Key Length Once an algorithm has been chosen, another parameter must be considered: the length of the key (also known as *secret*), which is also a factor in the security of the scheme. Once again, the main consideration is the trade-off between security and overhead: in general, longer



(a) The number of answers to how important the investigated parameter issues are



(b) The number of answers to whether the investigated parameter issues are difficult



(c) The number of answers to how frequent the participants encounter the investigated parameter issues in their own work

Figure 3: Survey results for the parameter related issues

keys offer greater security, but they also require higher processing power which can be very limited on a microcontroller. However, in order to judge which key size is sufficient for what level of security requires a basic understanding of the algorithm and its weaknesses.

Pa3 How to Implement a Freshness Mechanism In replay attacks, an attacker records a previous message which is encrypted or authenticated and resends it to achieve a particular goal. In order to avoid replay attacks, a *freshness* mechanism is used. This can be a monotonic counter or a timestamp added to the message, so that two otherwise identical messages will be different. There are several practical difficulties with freshness counters or timestamps. For one, a counter must be chosen appropriately large: once the counter wraps around, a new key must be used. It is also important that the counters are synchronized so that only particular counter values are accepted at particular times. Clock synchronization is a particularly tricky subject.

Survey Results Figures 3a to 3c show the survey results for the parameter issues. In general, there is consensus that these are important issues (Fig. 3a), but the responses on the difficulty are more nuanced. There is only slight agreement that choosing a suitable cryptographic algorithm is difficult, and there is strong disagreement that choosing a suitable key length is difficult (Fig. 3b). One participant remarked that for choosing parameters such as key length there are recommendations from NIST and AUTOSAR which simplify the process, a point we will return to in the recommendations in Section 5. We assume the availability of documented recommendations by trusted organizations is the reason for the perception of key length choice as an easy problem, and to a lesser extent for the algorithm choice. Another participant pointed out that the possibility for software updates, which are not universally supported in embedded systems, are of paramount importance to ensure continued security. For the most part, all three parameter choices happen rather infrequently (Fig. 3c).

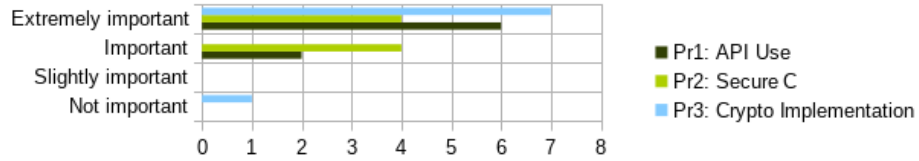
4.3 Programming

At some point, the chosen security measures must be implemented, and various pitfalls await the developer: from the correct use of APIs over implementation of cryptographic primitives to programming language pitfalls.

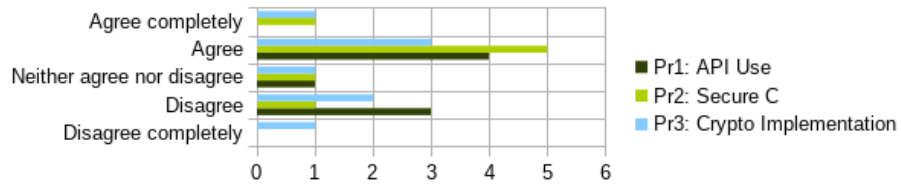
Pr1 Incorrect API Use One such source of difficulty for developing secure programs is the correct use of APIs. If the application programmer uses an API incorrectly, this may lead to insecure programs [1,2,5,11,14]. For instance, if programmers do not understand why an initialization vector is necessary, they may pass NULL for it, which may be allowed by the API but is semantically incorrect.

Pr2 Writing Secure C Code In addition to API specific problems, there are also typical development pitfalls that apply to any program written in the C programming language. It is easy to write insecure code, e.g., due to faulty memory management, pointer handling or lack of input validation [18]. These problems are well known and well documented, and yet they still occur in practice [21,22]. Since most automotive software is developed in C, writing secure C code is another programming issue.

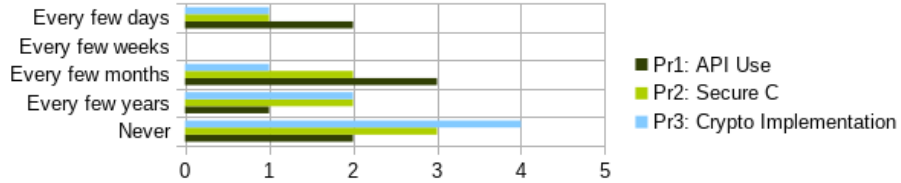
Pr3 Implementing Cryptographic Primitives or Libraries In order to implement security measures that use cryptography, cryptographic libraries or primitives must be available. However, since automotive systems are highly heterogeneous and often use minimal libraries, there is a chance developers may be tempted to implement their own cryptographic primitives. Consider for example the AUTOSAR standard for automotive software: AUTOSAR defines interfaces to access cryptographic libraries, but the standard also clearly states that the underlying implementation is the responsibility of the software vendor. Cryptographic libraries should always be written by cryptographers or security experts, otherwise there is a high probability that they are insecure [2,11]. If this



(a) The number of answers to how important the investigated programming issues are



(b) The number of answers to whether the investigated programming issues are difficult



(c) The number of answers to how frequent the participants encounter the investigated programming issues in their own work

Figure 4: Survey results for the programming related issues

Table 1: Responses per survey participant (see legend below), in the context of their own work

	Primary	Experience ³								
	Work Tasks	(Auto., Security, Safety)	D1 ⁴	D2 ⁴	Pa1 ⁴	Pa2 ⁴	Pa3 ⁴	Pr1 ⁴	Pr2 ⁴	Pr3 ⁴
Participant 1	Management	5,4,5	5,4,3	5,3,2	4,4,2	4,4,2	4,3,2	4,4,3	5,4,3	4,4,2
Participant 2	Industrial Research	5,4,1	3,3,3	3,4,2	2,3,2	1,3,2	4,4,2	2,4,2	4,3,2	2,4,2
Participant 3	Architecture	5,4,4	2,4,3	5,4,2	4,4,3	3,3,2	3,3,1	4,4,1	2,3,1	3,4,1
Participant 4	Architecture	4,5,1	5,4,5	5,4,2	3,4,4	3,4,3	2,3,2	2,3,5	4,3,1	5,4,1
Participant 5	Requirements Eng.	5,4,1	5,3,5	5,4,2	3,3,3	2,2,2	4,2,2	3,3,3	3,3,2	1,1,1
Participant 6	Req. Eng., Arch.	3,5,5	2,4,5	4,4,3	4,4,3	2,4,3	4,4,3	2,4,1	4,4,1	2,4,1
Participant 7	Req. Eng. Arch., Dev.	3,5,1	4,4,5	5,4,4	5,4,5	3,3,4	5,4,5	4,4,5	4,4,5	4,4,5
Participant 8	Development	2,3,1	4,4,5	4,4,4	3,4,3	3,4,4	4,4,3	4,4,3	4,4,3	4,4,3

	Experience	Difficulty	Importance	Frequency
Legend	1 Never	Disagree completely	Not important	Never
	2 Less than 1 year	Disagree	Slightly important	Every few years
	3 1 – 3 years	Neither agree nor disagree	Important	Every few months
	4 3 – 5 years	Agree	Extremely important	Every few weeks
	5 More than 5 years	Agree completely		Every few days

is not immediately obvious, consider the Debian SSL bug discovered in 2008: two small, superficially harmless changes by the Debian maintainers significantly lowered the entropy during SSL key generation, which led to a huge number of insecure keys. Thus, the implementation of cryptographic primitives or libraries is the final programming issue we consider.

Survey Results Figures 4a to 4c show the survey results for the programming related issues, some of which are surprising. Figure 4a shows an outlier for the importance of correct cryptographic implementations. However, since the context is the person’s own work, it may simply be that the person never works with cryptography, and thus finds it unimportant. There is consensus that it is an important issue in the other contexts, thus supporting this assumption.

Most surprising is that only half of the respondents agree that implementing cryptographic primitives is a difficult problem (Fig. 4b). We expected complete agreement here. Two people strongly stated in the comments that you should never implement your own “crypto”, which may be a hint for the reasoning behind the results: if you outsource it, it is not difficult. However, even in a general context several people answered that this is not difficult. Another possible reason may be hidden in the frequency (Fig. 4c): half of the experts answered that they never implement cryptographic primitives in their own work.

Table 2: Identified issues

D1	Identification of threats
D2	Choice of a key distribution mechanism
Pa1	Choice of suitable cryptographic algorithms
Pa2	Choosing a suitable key length
Pa3	How to implement a freshness mechanism
Pr1	Incorrect API use
Pr2	Writing secure C code
Pr3	Implementing cryptographic primitives or libraries

4.4 Intra and Inter Question Correlations

So far we have only considered aggregate survey results, but it may also be of interest to look at the participants' individual answers to investigate possible correlations. We will only highlight a few observations here.

Table 1 presents the dataset (in the context of each person's own work) in a codified form, and Table 2 summarizes the eight identified issues for easy cross reference with Table 1. Each row in Table 1 corresponds to the answer of one survey participant, and each answer is represented by one number, grouped in triplets. For experience, the triplet represent the answers for automotive, security and safety experience, respectively. For the eight issues, the triplet represents the answers for difficulty, importance and frequency, respectively. For instance, participant 1, who primarily works with management, has more than 5 years of experience with both automotive systems and safety, and 3 - 5 years of experience with security.

Studying this data, several interesting observations can be made. For instance, not one participant has more than 5 years of experience with both automotive systems *and* security, hinting at the fact that security is still relatively new in the automotive industry. It is also worth pointing out that the three participants with the most security experience collectively answered 20 times that the issues are extremely important, and 4 times that they are important, indicating a strong agreement with our claims, averaging at 3.835. For the remaining five participants, the average is 3.5, still a high level of agreement. Another, perhaps unexpected, observation is that, compared to the participants with less automotive experience, more experience in the automotive industry is negatively correlated to the importance participants ascribe the issues. This may be a side effect of the first observation, i.e., that participants with less automotive experience have more experience with security. Either way, the averages are still high, 3.375 for the group with extensive automotive experience (*auto_exp* = 5), and 3.875 for the group without (*auto_exp* < 5). Similar analysis shows that the more frequently a person is involved in a particular activity, the more difficult and important they rate that activity.

³Triplets in the order: Automotive, Security, Safety

⁴Triplets in the order: Difficulty, Importance, Frequency

5 Recommendations

As we have seen, security is a pervasive design issue which affects every level of the development process, and even trivial systems can be dangerous when exploited by an attacker. Consequently, security must be included in all development steps. Based on the previously identified issues, we give four recommendations how automotive software development can be made more secure.

R1 *Improve documentation, for instance by adding look-up tables for recommended key lengths, algorithms, MAC length and freshness parameters.*

Rationale: Software developers in the automotive industry are usually well trained in safety, but they often have little or no training in security. As a result, they may inadvertently introduce security relevant bugs into their code. Therefore, it should be made as easy as possible for automotive software developers to write secure code. This can in part be achieved through improved documentation. For instance, the difficulties in choosing the right key length, choosing the right cryptographic algorithm and choosing good parameters to guarantee freshness can be alleviated by adding more security related documentation.

R2 *The vehicle manufacturer should develop a process for key management.*

Rationale: The topic of key management deserves special attention, because of its wide ranging implications. If symmetric keys or a public key infrastructure setup are chosen, the vehicle manufacturer must maintain a central infrastructure to store and retrieve those keys on demand in a secure manner. The key management also needs to be coordinated with suppliers to clarify how and when the keys are installed. Furthermore, the keys must be accessible to licensed workshops for repair and maintenance.

R3 *Every development team should have access to at least one security expert and every team should have at least one developer who is trained in security.*

Rationale: Some of the identified issues can be alleviated by providing developer training or by providing access to security or cryptography experts. For example, for identifying threats at the architectural stage, a security expert should be available to provide an analysis. For the implementation of cryptographic libraries, cryptographers should be used, and developers should confirm that the library they use was developed by experts. Finally, API misuse can obviously be limited through developer training, too.

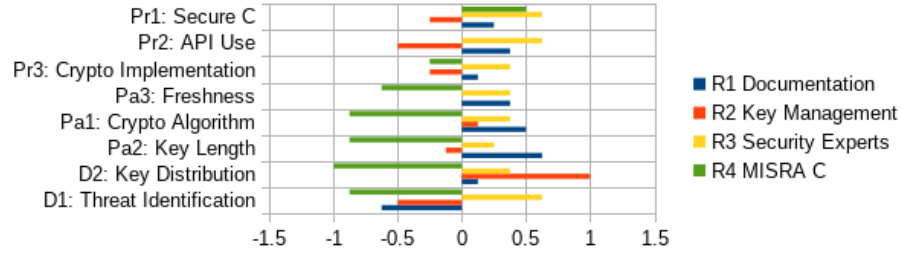


Figure 5: Answers to what extent the recommendations address or mitigate the various issues: positive values indicate a high extent of mitigation, whereas negative values indicate a low extent of mitigation

R4 Adhere to the MISRA C guidelines.

Rationale: The MISRA C guidelines were developed specifically to make the C programming language safer to use in critical systems.⁵ One effective result of requiring conformance to MISRA C is that all unsafe C library functions are implicitly forbidden to be used in production code. There are several commercial compilers which check MISRA C code compliance, but MISRA C contains many items which can not be checked automatically, or which require additional formal verification tools. Moreover, bugs which lead to security vulnerabilities can still happen. For example, it is possible to allocate a fixed-size buffer and accidentally write beyond its boundaries due to missing or insufficient run-time checks. Nevertheless, adherence to the MISRA C guidelines strengthens the security of the code considerably, even more so in combination with formal verification tools.

Survey results: In order to validate the recommendations, for each recommendation the survey also included the question to which extent it addresses the issues discussed earlier. The results are summarized in aggregated form in Fig. 5: if a majority answered “Not at all” or “Slightly”, the issue is depicted with a negative value, and if a majority answered “Significantly” or “Completely”, the issue is depicted with a positive value. The results are not surprising. Since R2 (key management) and R4 (MISRA C adherence) address very specific issues, they are only of value in those particular circumstances, whereas R1 (improve documentation) and R3 (security experts) help with almost all of the issues. This also echoes some of the findings of Acar et al. [1].

⁵There are several other coding guidelines for embedded, safety-critical or secure software, such as the *JPL C Coding Standard*, the *SEI CERT C Coding Standard*, or *The Power of 10 - Rules for Developing Safety Critical Code*, but a more detailed discussion is out of scope for this paper.

6 Related Work

As outlined in the introduction, interest in automotive security has been slowly on the rise for the last 15 years. Wolf, Weimerskirch and Paar [24] pioneered an initial analysis, and Koscher et al. [10], Checkoway et al. [4], and Miller and Valasek [13] demonstrated practical attacks, both local and remote. An added difficulty stems from the safety-critical nature of automotive engineering and the necessary integration of safety and security [3,6,8,12,16,25]. Studnia et al. [20] wrote a survey summarizing many automotive security issues. Similarly, we highlight commonly encountered security issues, but we additionally investigate how security experts perceive them.

7 Conclusion

With the increased communication of cyber-physical systems, securing software is of ever-increasing importance. Even systems which are generally perceived to have no safety-critical components can pose dangers when exploited by an attacker. An implication is that the interplay of safety and security must be examined closer; traditional views of safety may no longer be adequate.

The results of our survey with automotive security experts indicate that three of the eight issues we discussed are of particular interest. According to our survey, key distribution is a very important problem that is also very complex and it should be further investigated. Similarly, the choice and implementation of a mechanism for freshness is also an important and challenging problem. Both of these problems have been very well covered by academic research in the last 30 years, so it may be slightly counter-intuitive that they are still difficult to solve in the automotive context. However, this can be explained with the strongly constrained requirements for such systems. Threat identification on the other hand is an interesting problem because it is both an important and a frequent activity. Since it is a very dynamic activity that is strongly dependent on the concrete system under review, it will likely remain very important.

Some of the results may not be particularly surprising, but we believe it is still of value to formally document them in form of this survey. Conversely, some results were surprising, e.g., that several experts did not consider cryptographic implementations difficult.

Automotive software developers are typically well trained in addressing safety requirements, but writing secure software requires additional knowledge and skills. Consequently, new frameworks, platforms and standards should make it easier to write secure code, and they should foster an environment which supports secure development. As the survey results indicate, this can be partially achieved with supporting documentation to facilitate informed choices about security measures. However, improved documentation alone is not enough. In order to integrate security into the entire development process, cultural and organizational changes are needed. For instance, as the survey results illustrate, having ready access to security experts should alleviate many of the issues.

In order to achieve such a security conducive environment, several aspects must come together. First and foremost, there must be organizational support. Secure development can not be done without a security budget. Then there are the complex interactions of OEMs and suppliers which must be coordinated. More documentation how to securely use existing security functions should be added. Moreover, development processes must be adapted to include security reviews and security testing. All of the above entails a cultural change, so a concerted effort of all involved partners in the automotive industry is needed to secure future vehicles. Finally it can be noted that much of this discussion probably extends to embedded system development in other domains as well.

Acknowledgments

The research leading to these results has been partially supported by VINNOVA, the Swedish Governmental Agency for Innovation Systems, through the project “HoliSec” (2015-06894), and by the Swedish Civil Contingencies Agency (MSB) through the project “RICS”.

References

1. Acar, Y., Backes, M., Fahl, S., Garfinkel, S., Kim, D., Mazurek, M.L., Stransky, C.: Comparing the usability of cryptographic APIs. In: Proceedings of the 38th IEEE Symposium on Security and Privacy (2017)
2. Anderson, R.: Why cryptosystems fail. In: Proceedings of the 1st ACM Conference on Computer and Communications Security. pp. 215–227. CCS ’93, ACM, New York, NY, USA (1993)
3. Avizienis, A., Laprie, J.C., Randell, B., Landwehr, C.: Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing* **1**(1), 11–33 (2004)
4. Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S., Koscher, K., Czeskis, A., Roesner, F., Kohno, T.: Comprehensive experimental analyses of automotive attack surfaces. In: Proceedings of the 20th USENIX Security Symposium. pp. 77–92. San Francisco, CA, USA (Aug 2011)
5. Fahl, S., Harbach, M., Perl, H., Koetter, M., Smith, M.: Rethinking SSL development in an appified world. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security (CCS). pp. 49–60. ACM (2013)
6. Firesmith, D.G.: Common concepts underlying safety security and survivability engineering. Tech. Rep. CMU/SEI-2003-TN-033, Software Engineering Institute - Carnegie Mellon University (Dec 2003)
7. Islam, M.M., Lautenbach, A., Sandberg, C., Olovsson, T.: A risk assessment framework for automotive embedded systems. In: Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security. pp. 3–14. ACM (2016)
8. Jonsson, E.: Towards an integrated conceptual model of security and dependability. In: Availability, Reliability and Security, 2006. ARES 2006. The First International Conference on. pp. 646–653. IEEE (2006)
9. Koopman, P.: Embedded system security. *Computer* **37**(7), 95–97 (2004)

10. Koscher, K., Czeskis, A., Roesner, F., Patel, S., Kohno, T., Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S.: Experimental security analysis of a modern automobile. In: Security and Privacy (SP), 2010 IEEE Symposium on. pp. 447–462 (May 2010)
11. Lazar, D., Chen, H., Wang, X., Zeldovich, N.: Why does cryptographic software fail?: A case study and open problems. In: Proceedings of 5th Asia-Pacific Workshop on Systems. pp. 1–7. APSys '14, ACM, New York, NY, USA (2014)
12. Line, M., Nordland, O., Røstad, L., Tøndel, I.: Safety vs. security. In: Probabilistic Safety Assessment and Management (PSAM), Proceedings of the 8th international Conference on. pp. 685–699. IAPSAM (2006)
13. Miller, C., Valasek, C.: Remote Exploitation of an Unaltered Passenger Vehicle. Tech. rep., Defcon 23 (Aug 2015), <http://illmatix.com/Remote%20Car%20Hacking.pdf>
14. Myers, B.A., Stylos, J.: Improving API usability. Communications of the ACM **59**(6), 62–69 (2016)
15. Nowdehi, N., Lautenbach, A., Olovsson, T.: In-vehicle CAN message authentication: An evaluation based on industrial criteria. In: Vehicular Technology Conference (VTC-Fall), 2017 IEEE 86th. pp. 1–7. IEEE (2017)
16. Piètre-Cambacédès, L., Chaudet, C.: The SEMA referential framework: avoiding ambiguities in the terms "security" and "safety". International Journal of Critical Infrastructure Protection **3**(2), 55–66 (2010)
17. SAE International: SAE J3061_201601 - Cybersecurity Guidebook for Cyber-Physical Vehicle Systems (Jan 2016)
18. Seacord, R.C.: Secure Coding in C and C++. Pearson Education (2005)
19. Stevens, M., Bursztein, E., Karpman, P., Albertini, A., Markov, Y., Bianco, A.P., Baisse, C.: Announcing the first SHA1 collision (2017), <https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html>, February
20. Studnia, I., Nicomette, V., Alata, E., Deswarte, Y., Kaaniche, M., Laarouchi, Y.: Survey on security threats and protection mechanisms in embedded automotive networks. In: 2013 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W). pp. 1–12 (2013)
21. Szekeres, L., Payer, M., Wei, T., Song, D.: SoK: Eternal War in Memory. In: Security and Privacy (SP), 2013 IEEE Symposium on. pp. 48–62 (May 2013)
22. Van der Veen, V., Dutt-Sharma, N., Cavallaro, L., Bos, H.: Memory errors: the past, the present, and the future. In: Proceedings of the 15th International Symposium on Research in Attacks, Intrusions, and Defenses (RAID). pp. 86–106. Springer (2012)
23. Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full SHA-1. In: Shoup, V. (ed.) Advances in Cryptology – CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14–18, 2005. Proceedings. pp. 17–36. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
24. Wolf, M., Weimerskirch, A., Paar, C.: Security in automotive bus systems. In: Proceedings of the Workshop on Embedded Security in Cars (ESCAR) (Nov 2004)
25. Zalman, R., Mayer, A.: A secure but still safe and low cost automotive communication technique. In: Proceedings of the 51st Annual Design Automation Conference. pp. 1–5. DAC '14, ACM, New York, NY, USA (2014)