Quantitative Security Estimation Based on Safety Architecture Design Patterns

Christopher Preschern, Nermin Kajtazovic, Andrea Höller, and Christian Kreiner

Abstract—The increasing connectivity of embedded systems requires more attention to security aspects. Security should not be post-engineered to a system, but should already be considered during system design. However, especially during early design phases it is difficult to judge the impact of high level design decisions, such as the decision for an overall system architecture, on security.

To provide guidance for the system architecture selection for safety-related systems, we propose a quantitative security assessment method based on the application of design patterns. Based on security threats included in the patterns, we calculate a security metric to estimate the patterns' security influence for a specific system. We describe the calculation and application of the proposed security metric with an industrial case study.

Index Terms—Design patterns, metrics, safety, security.

I. INTRODUCTION

Security plays an increasingly important role for embedded systems as they become more and more interconnected. Incidents such as the Stuxnet attack [1] show that even safety-critical systems considered to be secure if not connected to the Internet can become subject to attacks. For safety-critical systems, security often does not play a primary role but is post-engineered after the safety-critical system is designed. An example for this is automation systems which partially have to use standardized insecure protocols. If these systems have to be accessed remotely, a common solution would be to provide a VPN connection and to tunnel the insecure protocol via this connection. However, such systems do not provide any defense in depth and if the VPN security measure can be broken or circumvented, an attacker can influence safety-critical functionality of the system. To analyze and counter relevant attacks and to minimize the effect of successful attacks, security should be considered during system design already. However, especially during early system design, such as the selection of an appropriate architecture meeting the system’s safety demands, it is difficult to estimate the effect of these high level design decisions on the resulting system security.

To provide security guidance for the architecture selection for safety-critical systems, we propose a metric to compare the safety of security architecture design patterns. The metric is based on the relevant set of security threats for a pattern and on their probability of occurrence. In this paper we describe with a case study how to calculate and use the security metric in order to choose a system architecture based on design patterns which satisfies both, safety and security needs.

This paper is structured as follows. Section II covers related work on security metrics with focus on security metrics for design patterns. Section III provides basics on Goal Structuring Notation and safety architecture design patterns. Both will be used in Section IV where we propose a security metric and apply it to a case study to select a system architecture from a set of safety architecture patterns. Section V concludes this work.

II. RELATED WORK

This section presents related work on general state of the art security metrics and in particular on security metrics for design patterns.

A. Security Metrics

Security metrics qualitatively or quantitatively describe the level of security for a system. Literature provides over 900 different security metrics [2]. Some of them are well-established metrics even used in industry and some of them are rather experimental and just described and applied in scientific papers. Several papers give an overview of existing security metrics and try to categorize them. For example, in [3] the authors list existing metrics and divide them into metrics which address a set of security targets or activities, metrics which relate to vulnerabilities, and metrics related to (risk) management. Alternatively security metrics can be categorized according to the main security attributes (e.g. confidentiality) which they address [4]. Another form of overview of existing security metrics which determines attributes such as correctness or measurability to qualify good security metrics is presented in [5]. The authors of [6] suggest a security metric classification based on the software development phase measured by the metric. They obtain the following classes of security metrics (for which we here show representative examples based on [6]):

- **Requirements Phase**: Metrics describing the fulfilled security requirements or the number of security requirements or misuse cases [7], [8]
- **Design Phase**: Metrics describing design decisions or attack surface related metrics [9], [10]
- **Implementation Phase**: Metrics describing the number of software failures and in particular security related failures or metrics describing the number of implemented security exception [11].
- **Testing Phase**: Ratio or number of security test cases.
reaction of the software to attack patterns [8]
• **Maintenance Phase:** Ratio of software changes due to security incidents; mean time between security incidents [7]
• **System-Level Metrics:** Some security metrics do not just focus on one software development phase, but rather include more than one or even all of them. An example for such an agent is the Common Criteria security certification. The certification assigns security levels to describe the achieved security of a product which can already be seen as a metric. To provide more detailed metrics based on the standard; the authors of [12] use specific Common Criteria requirements and analyze their achievement in a specific product to describe the product’s level of security.

**B. Pattern Security Metrics**

The authors of [13] evaluate the security of an architecture by considering different misuse patterns. They propose to analyze how many misuse patterns for an architecture can be countered when adding security patterns to improve the architecture. The calculated value then represents the level of security for the applied security patterns. This idea is taken one step further in [14], where the patterns present in an architecture are first automatically mined and then evaluated regarding their influence on security.

The authors of [15] present a similar approach. They map security patterns to quality attributes (confidentiality, availability ...) which these patterns achieve or improve and estimate the influence of the patterns on these quality attributes. During system design, threat trees are constructed and the risk of the threats is estimated. Patterns which can reduce these risks are then selected. In another paper [16] the authors focus on determining the risk reduction metrics and other security metrics for the patterns.

Halklidis et al. provide the related work which is closest to our proposed security metrics for safety patterns. They analyze a set of security patterns and assign values representing their suitability to counter STRIDE threats [17], [18]. These patterns are then taken into consideration when designing a secure architecture. First, they determine possible attacks for a system and structure them in fault trees. Next, they estimate probabilities for the attacks and decide for a set of security patterns by considering how much the attack probabilities can be reduced by the different patterns [19]. Compared to our work, they use patterns from a different domain (not safety) and they do not implicitly cover the threat impact in the pattern structure as it is the case in our work.

With GSN it is possible to link a high level goal to very specific and detailed information/evidence which supports the high level claim/goal.

Fig. 1 shows an explanatory example for a GSN diagram. In order to show that the rather general top-level goal (“The system is safe in case of hardware faults”) is achieved, it is broken down into more specific sub-goals and linked to these sub-goals and their evidence of achievement with arrows. First, for the specific example, an architectural measure (GSN strategy element: “Redundant Hardware & Output Voting”) is implemented to achieve the top-level goal. Next, sub-goals relevant to achieve the architectural measure come up (e.g. “Voter works properly”). If these sub-goals are specific enough, they can be linked to GSN evidence elements (e.g “Voter test documentation”). Otherwise they could be further split up. If all sub-goals are related to evidence elements, one gets a complete argument for the achievement of the top-level goal. Further information and examples about GSN can be found in [20].

**B. Safety Architecture Patterns**

Design patterns describe good solutions for re-occurring problems. There are patterns for software design or architectures in general and also patterns which focus on more specific topics. We will focus on the safety architecture patterns from [21] which provide solutions for high level software or hardware architectures for safety-critical systems.

An example for such an architecture is the TRIPLE MODULAR REDUNDANCY (TMR) pattern which uses three identical channels and a voter which decides for the majority of the channel outputs. This architecture increases the system’s availability and protects from random hardware faults.

The 15 patterns in [21] all contain the following:
• **Description in which Context they can be applied**
• **Description which Problem the solve**
• **Description of the Solution to that problem including a diagram showing the system’s main components**
• **Description of the Consequences when applying the pattern**
• **GSN diagram containing security threats**

We will especially focus on the GSN diagram of the patterns which provides a structured presentation of safety-relevant threats. The threats were obtained with the STRIDE threat modeling approach which is described in detail in [22]. The GSN diagrams consist of a collection of goals who’s aim is the prevention of attacks related to
safety-critical STRIDE threats.

IV. CASE STUDY: APPLYING THE SAFETY PATTERNS AND ESTIMATING THEIR EFFECT ON SECURITY

In this section we describe an industrial case study for which we consider different safety architectures and compare their security. Fig. 2 gives an overview of this section and shows how we will calculate a security metric to guide the architecture selection.

A. System Description

Our case study is an automation system for hydro-power plants. The automation system has to control turbines and has to detect critical states such as too high voltages in the power plant generator. Malfunctions of the hydro-power plant controller could lead to damage of the machinery such as the turbines and could even be a threat to human safety for people operating the power plant. Therefore, the specific system has to be safety-integrity-level 3 (SIL3) certified according to the IEC 61508 safety standard. This standard recommends using redundant hardware for SIL3 systems. Therefore, we will just consider safety patterns containing redundant hardware as candidates for the system architecture. Additional hardware constraints for the hydro-power plant controller further narrow down the applicable patterns. For the controller hardware design, at most three independent complex hardware elements (programmable microcontrollers) are available due to cost reasons. Thus, just patterns containing at most three independent hardware channels will be considered.

The above mentioned constraints give us the choice of the following four safety architecture patterns (detailed architecture descriptions can be found in [21]):

Homogenous Duplex, Heterogenous Duplex

The architecture (Fig. 3) consists of two homogenous (identical) or heterogenous (diverse) channels which compute outputs from input data. A switch decides which output is actually taken. Normally, the primary channel is used. However, if the fault detector detects a failure in the primary channel, the switch takes the output of the backup channel.

For the used hardware, the two channels would be implemented on one microcontroller each. One microcontroller would be used to implement the fault detector. The switch would be realized with additional hardware.

Heterogenous Triple Modular Redundancy, Homogenous Triple Modular Redundancy

The architecture consists of three homogenous (identical) or heterogenous (diverse) channels. All channels compute an output and a voter decides for the majority of the outputs.

From a safety point of view, all of the four architectures above are applicable for the hydro-power plant controller. Now, we want to evaluate their influence on security to get some more guidance for the architecture selection.

B. Security Threats

The safety architecture patterns of [21] come with a GSN diagram representing safety-relevant threats which were gathered for the patterns with a STRIDE analysis [22]. Fig. 5 shows the GSN diagrams for the four considered patterns. We can see that some sets of specific threats (smaller font size in the Fig.) for the different patterns are similar, but the combination of the threats to an overall security argument (larger font size) is very different. For example, the difference between the heterogenous and the Homogenous TMR pattern is that for the heterogenous version, 2 of 3 single channels have to be secure of attacks, while for the homogenous version, no GSN multiplicity element is present in the diagram. This is, because if the channels are identical, an attacker usually can easily attack all three channels as soon as he knows how to compromise one of them. Therefore, the security GSN diagram just considers one channel for the Homogenous TMR pattern.

C. Security Risk Estimation

We have seen in Fig. 3 that many of the threats are similar for the different patterns. We now rate these general threats without yet knowing which specific architecture will be chosen. We conducted a meeting with system architects and security experts of our industrial partner to estimate the probability of attacks related to specific threats for the hydro-power plant controller. Note that we do not estimate the attack impact (which is common in security risk estimation like [22]), because the impact is already implicitly covered in the position of the threat in the GSN diagram. This is an advantage of the presented approach compared to pattern security metrics described in literature. The results of the probability of attack estimation made by the industrial partner is shown in Table I where the introduced “Goal Confidence” (probability that the threat-related GSN goal is achieved) is the converse probability to the estimated “Probability of Attack” and the values (none, low, medium, high) are substituted with (0.00, 0.01, 0.05, 0.20). These values were chose based similar estimations in literature [15], [19].
TABLE I: GENERAL THREATS FOR ALL CONSIDERED ARCHITECTURES

<table>
<thead>
<tr>
<th>Threat</th>
<th>Prob.of Attack</th>
<th>Goal-Conf.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Denial of Service, microcontroller output data</td>
<td>none</td>
<td>1.00</td>
<td>The outputs are always hardwired to the actors, switches or voters</td>
</tr>
<tr>
<td>Denial of Service of microcontroller input data</td>
<td>low</td>
<td>0.99</td>
<td>The inputs are received via a separate LAN</td>
</tr>
<tr>
<td>Tampering of microcontroller output data</td>
<td>none</td>
<td>1.00</td>
<td>The outputs are always hardwired to the actors, switches or voters</td>
</tr>
<tr>
<td>Tampering of microcontroller input data</td>
<td>low</td>
<td>0.99</td>
<td>The inputs are received via a separate LAN</td>
</tr>
<tr>
<td>Elevation of privilege on microcontroller</td>
<td>high</td>
<td>0.80</td>
<td>Controller connected to the local LAN; OS could be compromised</td>
</tr>
<tr>
<td>Microcontroller spoofing</td>
<td>medium</td>
<td>0.95</td>
<td>Users often just use weak passwords</td>
</tr>
<tr>
<td>Denial of Service of switch/voter output data</td>
<td>none</td>
<td>1.00</td>
<td>The outputs are hardwired</td>
</tr>
<tr>
<td>Tampering of switch/voter output data</td>
<td>none</td>
<td>1.00</td>
<td>The outputs are hardwired</td>
</tr>
<tr>
<td>Elevation of privilege on switch/voter</td>
<td>low</td>
<td>0.99</td>
<td>Unlikely, because it is a simple hardware element</td>
</tr>
<tr>
<td>Switch/Voter spoofing</td>
<td>none</td>
<td>1.00</td>
<td>The outputs are hardwired</td>
</tr>
</tbody>
</table>

D. Goal Confidence

With the “Goal Confidence” for the threat goals estimated in Table I, we can now compute an overall metric for the security of the pattern architectures. We simply use probability theory to propagate the goal confidence to higher level goals. Table II shows the calculation rules.

With the established rules we can now calculate the goal confidence of the top-level goals for the security GSNs of the four considered safety architecture patterns. First, we calculate the goal confidence for “Single Channel works properly” or “Fault Detector works properly” (goals from Fig. 5):

\[
G_{ch,fd} = \prod_{i=1}^{6} G_{mc\text{-}threats} = G_{mc\text{-}out-dos} \cdot G_{mc\text{-}in-dos} \cdot G_{mc\text{-}out-tampering} \cdot G_{mc\text{-}in-tampering} \cdot G_{mc\text{-}spoofing} \cdot G_{mc\text{-}exp} = 1.00 \cdot 0.99 \cdot 1.00 \cdot 0.99 \cdot 0.80 \cdot 0.95 = 74.5\%
\]

TABLE II: GOAL CONFIDENCE FORMULAS

<table>
<thead>
<tr>
<th>GSN Relation</th>
<th>Goal-Confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>N subgoals have to be met to meet the high-level goal:</td>
<td></td>
</tr>
<tr>
<td>( GC = \prod_{i=1}^{N} G_{sb_i} )</td>
<td></td>
</tr>
</tbody>
</table>

| M out of N subgoals have to be met to meet the high-level goal: |
| \( GC = \sum_{k=1}^{N} \left( \binom{N}{k} \prod_{i=1}^{k} (1 - G_{sb\_i}) \right) \) |

\( GC_{sv} = \prod_{i=1}^{4} G_{sv\text{-}threats} = G_{sv\text{-}dos} \cdot G_{sv\text{-}tampering} \cdot G_{sv\text{-}exp} \cdot G_{sv\text{-}spoofing} = 1.00 \cdot 1.00 \cdot 0.99 \cdot 1.00 = 99.9\% \)
Next we calculate the goal confidence for “Switch works properly” or “Voter works properly”:

Next we calculate the goal confidence for the top-level goals for our four considered architectures. We simply apply the equations from Table II according to the GSN diagrams from Fig. 3 to obtain the goal confidence values for the Homogenous TMR, Heterogenous TMR, and the Homogenous Duplex system.

\[
GC_{hom-mes} = GC_{sv} \times GC_{ch/fd} = 0.99 \times 0.74 = 73.7\%
\]

\[
GC_{het-mes} = GC_{sv} \sum_{i=3}^{3} 3 \left( GC_{ch/fd} \right)^{i-3} \left( 1 - GC_{ch/fd} \right)^{3-i} = 83.0\%
\]

\[
GC_{hom-duplex} = GC_{sv} \times GC_{ch/fd} = 0.99 \times 0.74 = 73.7\%
\]

For the Heterogenous Duplex system we first calculate the goal confidence of the “Fault Detector and one Channel work properly” goal and then the overall goal confidence.

\[
GC_{fd-and-ch} = GC_{ch/fd} \left( \sum_{k=3}^{2} \frac{2}{k} \left( GC_{ch/fd} \right)^{k} \left( 1 - GC_{ch/fd} \right)^{k-3} \right) = 0.74 \times 2 \times 0.74 - 0.74^2 = 69.7\%
\]

\[
GC_{het-duplex} = GC_{sv} \left[ 1 - \sum_{j=3}^{2} \left( 1 - GC_{sub} \right) \right] = 0.99 \times \left[ 1 - \left( 1 - GC_{ch/fd} \right)^2 \left( 1 - GC_{fd-and-ch} \right) \right] = 85.7\%
\]

E. Architecture Decision

Based on the probability estimation for threat-related attacks, we now have the following security metrics for our four considered architectures:

- Homogenous TMR: 73.7%
- Heterogenous TMR: 83.0%
- Homogenous Duplex: 73.7%
- Heterogenous Duplex: 85.7%

These values represent the probability of an attacker tampering with the safety functionality of the system. However, one has to be careful when interpreting these values. The absolute values are highly dependent on our mapping of the (none, low, medium, high) terms used during the threat probability estimation to quantitative values. This means that the absolute values are not well suited to directly represent the exact probabilities for attacks. However, when comparing the values for the different architectures, the calculated security metrics can be very useful. For example, we can see that the Homogenous TMR pattern for our case study is less secure than the Heterogenous TMR pattern. This is not very surprising, because obviously it is more difficult for an attacker to attack three diverse systems compared to three identical systems. Still, for other architecture pairs, the security metrics do give valuable information. For example, the metrics say that the Heterogenous TMR pattern is less secure than the Homogenous TMR pattern in the case of our hydro-power plant controller with its specific threat probability estimation. This information is not obvious and can give guidance for preferring one safety architecture over another. It is also interesting to compare the security metrics of the patterns to the security metric of a single microcontroller (which would be the architecture for the system without applying any of the patterns):

- Homogenous TMR:
  \[
  \frac{GC_{hom-TMR}}{GC_{ch/fd}} = 73.7\% / 74.5\% = 0.99
  \]

- Heterogenous TMR:
  \[
  \frac{GC_{het-TMR}}{GC_{ch/fd}} = 83.0\% / 74.5\% = 1.11
  \]

- Homogenous Duplex:
  \[
  \frac{GC_{hom-duplex}}{GC_{ch/fd}} = 73.7\% / 74.5\% = 0.99
  \]

- Heterogenous Duplex:
  \[
  \frac{GC_{het-duplex}}{GC_{ch/fd}} = 85.7\% / 74.5\% = 1.15
  \]

These values represent the relative security improvement when applying the pattern. We can see that the homogenous version of the patterns actually decrease the overall security of the system. This is not surprising, because they increase the attack surface for the system. The heterogenous versions also increase the attack surface, but they also provide more security, because compared to a single microcontroller, they provide diverse systems which requires an attacker to break more than one of the microcontrollers.

For the hydro-power plant controller, the information about the influence of the architecture on security supported the decision to implement the Heterogenous Duplex pattern. Fig. 6 shows the high-level architecture of the resulting hydro-power plant controller.

F. Discussion

We calculated security metrics for four patterns; however, with the calculation rules given in this paper, it is no problem to calculate the security metrics also for the other patterns in [21]. This would be necessary, for example, if there were
V. CONCLUSION

In this paper we showed with a case study how security metrics can help to make decisions for early system architecture design. We estimated the probability of general attacks for a case study and calculated a resulting security metric when applying different safety architecture patterns in order to compare them and decide for one of them. Apart from the design guidance by the patterns as good solutions to specific problems, with the described approach a system designer now has the possibility to quantitatively estimate the influence of safety architecture selection on security. The presented method to compare the security of different architectures aids the decision for a specific architecture from a security point of view.

For future work it would be interesting to apply the security metric to other case studies or to apply the metric to GSN diagrams of other domains. The metric could even be extended to evaluate both, safety and security, of an architecture if appropriate safety GSN diagrams for the architecture are available. Another point for future work is to thoroughly evaluate the proposed security (for example with a sensitivity analysis) metric. At this point it is difficult to compare the metric to security metrics in literature, because the metrics in literature do not describe the security of general architectures and require more detailed security assumptions/information about the system (such as the impact and probability of detailed attacks). However, in the long term we plan to evaluate the proposed metric by analyzing the number of incidents on different systems which apply the safety architecture patterns and we expect to obtain a relation to our security metric.

With the approach described in this paper we provide a way to easily compare the level of security when applying safety design patterns. With this method we aim to increase the awareness and consideration of security problems in the safety domain.

REFERENCES


Christopher Preschern received the master's degree in telematics from Graz University of Technology in 2011, focusing on software product lines, automation systems and IT-security. He is working toward the Ph.D. degree in electrical engineering at the Institute for Technical Informatics, Graz University of Technology. His research focuses on design patterns and their relationship to non-functional quality attributes, in particular safety and security.
Nermin Kajtazovic received his master's degree in telematics from Graz University of Technology in 2011. He focused on variability management in component-based architectures for the automotive domain. He is a PhD student at the Institute for Technical Informatics, Graz University of Technology. His research is related to component-based systems in the safety domain.

Andrea Höller received her master's degree in telematics from Graz University of Technology in 2013. She focused on implementation and attacks elliptic curve cryptography hardware. Currently, she is a PhD student at the Institute for Technical Informatics, Graz University of Technology. Her research is related to multi-core technology, safety-critical systems, and diversity measures.

Christian Kreiner graduated and received the Ph.D. degree in electrical engineering from Graz University of Technology, in 1991 and 1999, respectively. From 1999 to 2007, he served as head of the R&D Department, Salomon Automation, Austria, focusing on software architecture, technologies, and processes for logistics software systems. He was in charge to establish a company-wide software product line development process and headed the product development team. There, he led and coordinated a long-term research program together with the Institute for Technical Informatics of Graz University of Technology. There, he currently leads the industrial informatics and model-based architectures group. His research interests include systems and software engineering, software technology, and process improvement.