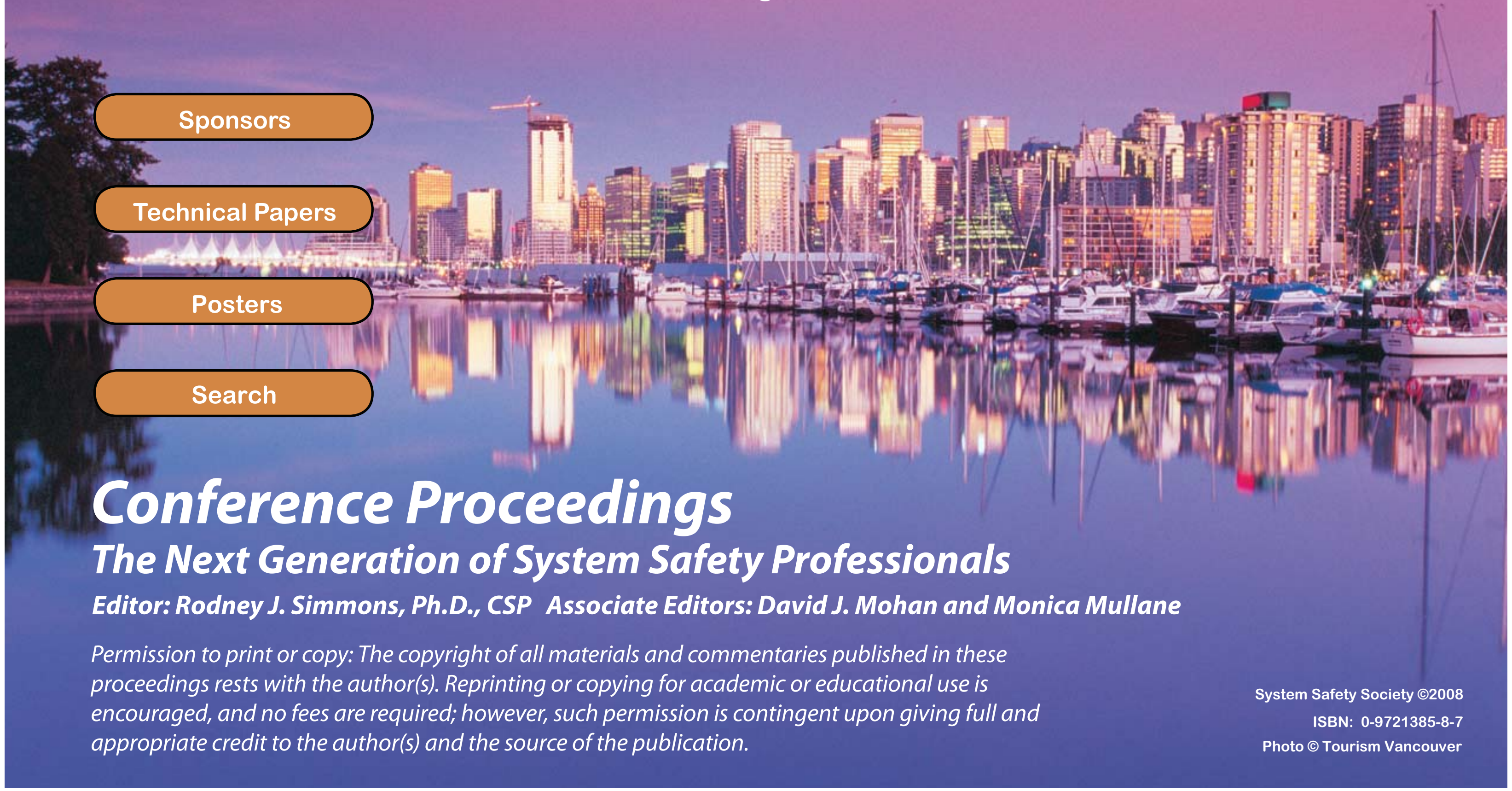
26th International System Safety Conference

Hosted by the System Safety Society • Sheraton Wall Centre • Vancouver, Canada

25 - 29 August 2008



Sponsors

Technical Papers

Posters

Search

Conference Proceedings

The Next Generation of System Safety Professionals

Editor: Rodney J. Simmons, Ph.D., CSP Associate Editors: David J. Mohan and Monica Mullane

System Safety Society ©2008

ISBN: 0-9721385-8-7

Photo © Tourism Vancouver

Embedded Security in Safety Critical Automation Systems

Thomas Novak, Ph.D.; Institute of Computer Technology; University of Technology, Vienna, Austria

Albert Treytl; Research Unit for Integrated Sensor Systems, Austrian Academy of Sciences,
Wiener Neustadt, Austria

Andreas Gerstinger; Institute of Computer Technology; University of Technology, Vienna, Austria

Keywords: life cycle, safety critical software, security, automation system

Abstract

Industrial and building automation systems are more and more important in industry and buildings. New services and novel fields of application call for dependable systems. Two very important properties of such a system are functional safety and system security. Today’s automation systems are lacking of efficient security features, even though promising extensions are available. Some have already been enhanced with safety features on application level. However, during integration dependencies between safety and security are not taken into consideration. Safety and security are examined independently. Therefore additional effort in terms of development costs, performance and resources of the system is necessary to receive a dependable automation system. Investigating security together with safety leads to a reduction of effort in the different phases of development. That is because they have some similar objectives, but realized by different measures. The intention of the paper is to evaluate the different measures and discuss them with regard to the safety and security impact as well as to show the associated benefit with special focus on building automation. Practical examples are taken from a real-world project called SafetyLon that strives for making LonWorks safe [3].

Introduction

Looking at safety and security historically these issues have been handled separately. Especially, safety systems are setup and operated totally disjoined from other systems. I.e., each domain had its own physically separated system and gateways are required to connect them. For safety systems additionally absence of reaction is required and has to be proven usually resulting in a limited read-only access to the safety system.

New trends such as remote access via the Internet, advanced control operations or cost reduction by using shared networks, not limited but obvious in the area of building automation, advise to rethink this separation and setup concepts for systems that allow common usage by safety, security, HVAC (heating, ventilation and air conditioning) and other control functions. These trends break up the isolated structure of existing networks and therefore introduce new risks and threats to working systems.

Today’s automation systems and networks are in general lacking efficient security features. If available security is an extensions that is seldom used and often has non-negligible drawbacks [1,2]. In the same way automation systems have been enhanced with safety features on application level [3]. What is in common is that dependencies between different operation modes, and safety and security in particular are not considered. Safety and security are examined independently not considering potential hazardous side effects on each other.

Using redundancies in building automation control systems by integrating safety critical, security relevant and normal operation into a single communication system would allow for big savings. In a first approach embedding security measures are to guarantee the correct execution of all safety relevant operations in a mixed operation environment.

This goal requires that the systems offer a flexible security framework that on the one hand handles the correct usage of resources needed by safety, e. g. QoS parameters like keep alive intervals, communication times, the implicit access control demanded by the safety application, and on the other hand also offers the same and other services like access rights, authentication to other applications.

This paper introduces a possible common approach investigating security together with safety that on the one hand leads to a reduction of effort in the different phases of development and on the other hand aims at the design of dependable systems which have security and safety in their core. The intention of the paper is to evaluate the different safety and security measures and discuss them with regard to the safety and security impact as well as to show the associated benefits with special focus on building automation. In the following security and safety goals with their commonalities are described, then a life cycle model that allows for a combined approach towards security and safety is introduced. Finally, rules for conflict resolution and a measure assessment are presented. Practical examples are taken from a real-world project called SafetyLon that strives for making LonWorks safe (SafetyLon [3]) and from the authors activities related to secure industrial and building automation systems.

Common Procedure: Integrity, Authentication and Authorization

In order to understand the potential of synergies given between safety and security the goals of the two areas should be analyzed: Safety goals are

1. Integrity, which demands the correct operation of the system under all defined circumstances within a fixed period of time. It is usually divided into stochastic (hardware) integrity and systematic integrity.
2. Authentication, which demands that a message is coming from the correct source. A common approach is source based addressing.
3. Availability is not necessarily a direct safety goal since a non-available system can find a simple fail-safe state by going to no-operation, yet for practical reasons this trivial solution is not desired.
4. Authorization is usually implemented implicitly by allowing authenticated operation. Additionally, a check for maximum plausibility is sometimes applied.

Security goals are

1. Confidentiality, meaning that only authorized entities must be able to read confidential data,
2. Integrity, stating that no unauthorized entity must be able to change data without being detected
3. Availability, mandating that data is on-hand when it is needed
4. Authentication, allowing to determine the sender/creator of a message
5. Authorization, defining access rights.
6. Non-Repudiation, giving evidence that the sender/creator of a message issued the message.

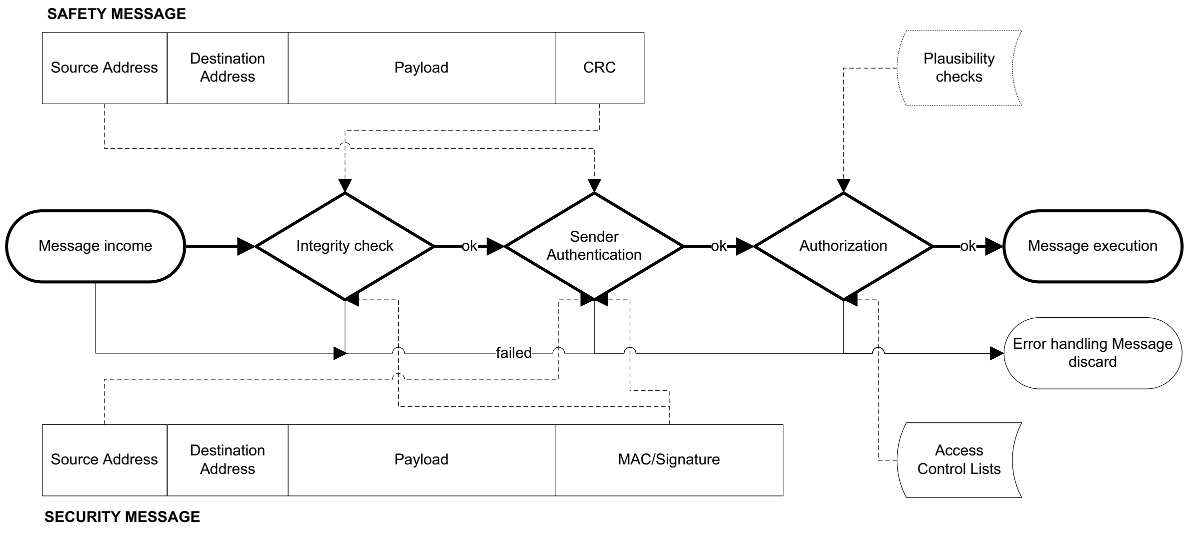


Figure 1 — Common procedure: integrity, authentication and authorization

Looking at the security goals relevant for automation systems confidentiality and non-repudiation can be neglected for the interesting systems in automation [1,2,4]. Hence, comparing the remaining important security goals – often abbreviated as CIA – with the safety procedure a common pattern between safety and security can be identified.

A chain starting with integrity verification, authentication followed by an authorization is given (see Figure 1). Measures in this chain are differently implemented since they aim at different sources of failures, but pursue the same goals; Security aims at protection and defense of threats from intentional attacks, safety measures are a protection against malfunction of the system itself including fault tolerance, functional safety integrity and fault resistance [5]. E.g., safety authorization is based on maximum plausibility, i.e. is the value within a fixed range, or even simpler allows everything the passes the authorization stage. In contrast, security demands fixed access control lists (white or black lists). Similar for the integrity, CRC codes computable by everyone are sufficient against stochastic failures, whilst security necessitates cryptography that requires the knowledge of a secret key to properly verify the integrity of a message.

Looking at the operational level a first approach would be to replace the measures with a common measure that fulfills the requirement for both domains, e.g. a cryptographic message authentication code (MAC) instead of the CRC. Yet, finding this commonalities is not that straight forward since measures can need different efforts or even find no common equivalent. However, the optimization goes much beyond this simple replacements; they need a holistic analysis that requires considerations of safety and security in the complete life cycle from development, to operation and disposal.

Life Cycle Model

A life cycle model is a structured and systematic model covering the *development* and *use* phase of a system. Within this paper a life cycle model is used as generic term for every procedure starting with a product's conception and ending with its disposal. Or, in other words a life cycle model specifies a logical activity flow of a project.

Typically, a life cycle starts with the concept phase including a definition of the scope, the purpose and examination of the environment. Next, a requirements analysis is performed and a specification is set up. It includes functional, performance, usability, reliability, supportability or design-constraints requirements. After that a design or architecture of the system is created that is realized in a further step. Subsequently, the product built is verified. *Verification* is the process of evaluating a system to determine if the products of a given development phase satisfy the conditions imposed at the start of that phase [6]. In other words, the question is, 'Are we building the product right?' Following the verification process, the system is installed and validated. *Validation* is the process of evaluating a system during or at the end of the development process to find out whether it satisfies the specified requirements [6] comprising its intended use. Validating a product is associated with the question, 'Are we building the right product?' In the use-phase, the system is in operation and has to be maintained. Finally, at the end the system is decommissioned.

Life cycle approaches: A life cycle approach has become 'best practice' in the safety domain. Examples of safety life cycles can be found in [7] or in the international standard for functional safety IEC 61508 [8]. There has been a common understanding that activities such as fault avoidance and fault control must be applied at different stages of the life cycle. Often safety assessment work has been confined to assessing whether the proposed architecture meets the target failure probabilities [9]. Less attention was paid to the installation, maintenance and disposal phase [10]. Therefore, a lot of serious safety problems occurred during that phases.

Similar approaches, albeit less detailed and accepted, have been development in the security domain. In [11] it is outlined that building any type of software securely is only possible if security issues are considered during all phases of the life cycle. Hence, seven so-called touchpoints (a set of best practices) are introduced, each of them applicable to a different life cycle phase. In the international standard IEC 15408 [12], also known as Common Criteria (CC), security related activities for the various life cycle phases are specified. They are organized in classes such as activities for requirement specification or installation. Additionally, CC specifies a way of how to derive security requirements. In general, the trend to ensure security during the various stages of a product life is clearly perceptible. Its importance is growing due to the increasing complexity and connectivity of security critical systems

Whereas in the safety domain life cycle models are specified, the security domain very often determines activities relating to security, but does not define a model, i.e. the flow or order of activities. The basic idea of the safety-security lifecycle presented in Figure 2 is to use the safety lifecycle from IEC 61508 and integrate the security approach specified in Common Criteria (CC). Requirements how to proceed are given for every phase of the system life. Moreover, activities are added to consider safety and security dependences resulting from integrating safety and security.

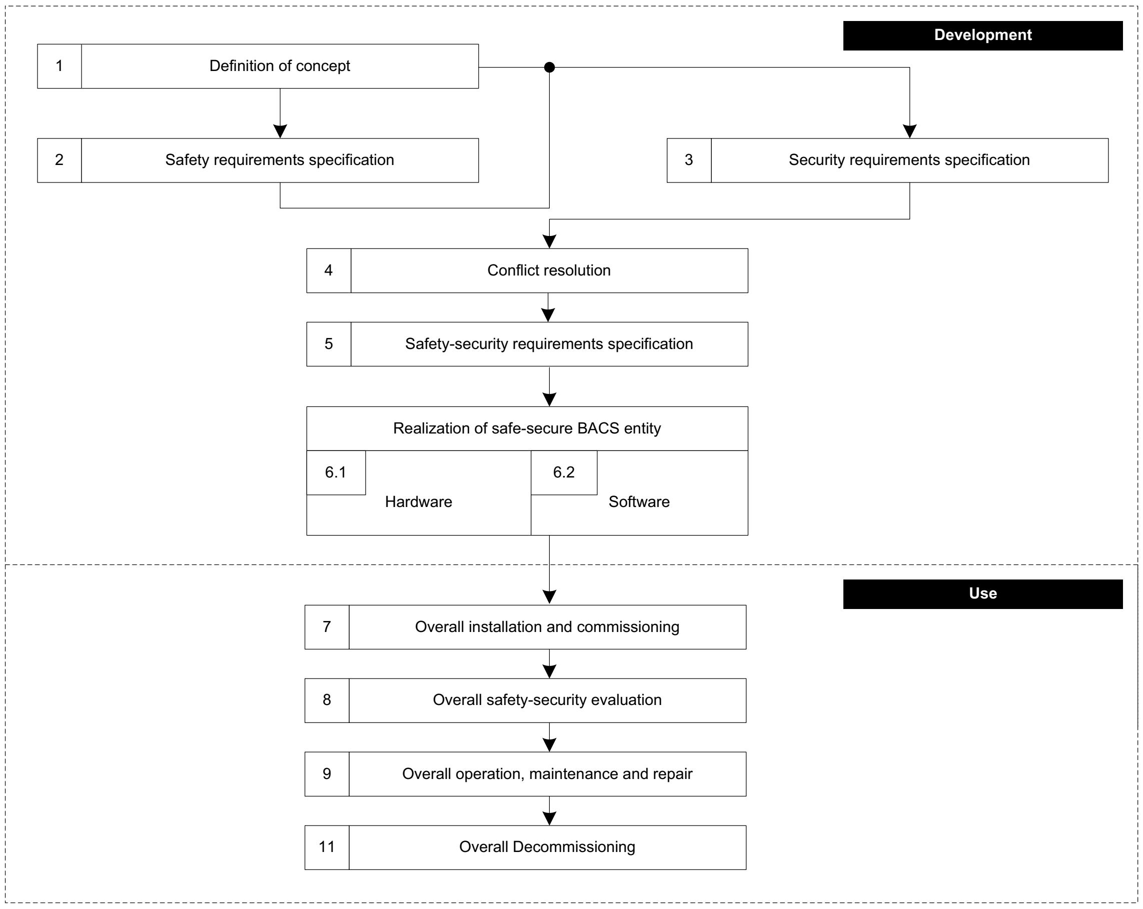

Development
1 Definition of concept
2 Safety requirements specification
3 Security requirements specification
4 Conflict resolution
5 Safety-security requirements specification
Realization of safe-secure BACS entity
6.1 Hardware
6.2 Software
Use
7 Overall installation and commissioning
8 Overall safety-security evaluation
9 Overall operation, maintenance and repair
11 Overall Decommissioning

Figure 2 — Safety-security life cycle model

The two international standards IEC 61508 and CC are not application specific. They provide a set of requirements and procedures to be used in different applications. They are chosen as basis of the proposed life cycle model because they have a good coverage of functional safety and security aspects. And the standards are considered to be well-accepted in their domains. In addition, both make a classification and a comparison of systems possible by specifying different levels: 4 safety integrity levels (SIL) for safety related and 7 evaluation assurance levels (EAL) for security related systems. The rigor and amount of requirements increases with higher levels.

1. The safety-security life cycle model is divided into two major parts: The first part includes activities related to the *development* of an entity in an automation system, e.g. a node. The development phase includes safety dependent and security dependent activities. In addition, an approach to deal with conflicts resulting from contradicting safety and security requirements is integrated into the development phase.
2. The second block is related to the *use* of the automation system. and refers to all activities during the use of the complete system.

Although activities of both phases are shown in a sequential way in Figure 2, the life cycle model only intends to show that activity *n+1* requires input of activity *n*. For the sake of readability recursions are not included in this figure and activities may be required to be redone during system life to receive a safe-secure system, e.g., conflict resolution (discussed later in the paper) may reveal a conflict that has an impact on the hardware environment. Hence, safety and security requirements have to be investigated again.

Development phase: The development phase is the first phase of the safety-security lifecycle model. Stages 1 and 2 are following IEC 61508, stage 3 is following the Common Criteria. Stages 4 to 5 are additional activities to handle dependencies between safety and security. Finally, stage 6 deals with the realization of an entity.

Definition of the concept: As mentioned in [13], the development phase begins with definition of the concept that is input to the safe dependent and the security dependent part. First of all the purpose and the scope of the automation system in general and its entities in particular must be defined. It is important to know what the automation system is used for, its field of application. Additionally, it is required to specify the scope: Are there 10000 or only 100 nodes in the automation system? Are they connected to an intranet or even to the Internet?

Safety requirements specification: The safety dependent part start with a safety scope definition where the boundaries of the entity in an automation network are determined. Next, a hazard and risk analysis is performed within the aforementioned boundaries. The hazards can result from failures on the network such as delay of messages or result from failures on the entity like memory failures. The hazards are identified and the risk coming from the hazards is assessed. If the risk is not acceptable, i.e. higher than the target safety level, safety functions must be specified to reduce the risk. E.g., a CRC implemented to detect stochastic failures. Requirements on such functions are the output of the safety dependent part comprised in the safety requirements specification.

Security requirement specification: Results from the safety investigation and definition of the concept are input to the security dependent part of the safety-security life cycle model. First, the security environment is examined: the physical environment and assets, i.e. information and resources that require protection. Typical examples of assets in automation systems are sensor or actuator data. Next, the assets are valued and threats to them are specified. A typical threat to sensor data is deliberate manipulation. Moreover, the risk resulting from a threat is estimated. The measures to reduce the risk or in other words requirements on the security functions required to protect the assets are specified according to the value of the asset and the risk of threat to the asset. E.g., to detect manipulation of sensor data a message authentication code (MAC) needs to be integrated.

Conflict resolution: Safety requirements and security requirements are investigated in the conflict resolution activity. Interaction between the different kinds of requirements is examined. Conflicts between safety and security requirements are resolved by a conflict resolution policy presented in the following section.

Safety-security requirements specification: In case of having solved the conflicts, a safety-security requirements specification is available. It is noteworthy that from this stage on there are not any safety and security requirements, each belonging to either safety *or* security, but there are only safety-security requirements. There are requirements on the functions and measures to be implemented on an entity, requirements on the hardware and software design

Realization of a safe-secure entity: The activity is separated into hardware and software realization as it is common practice and suggested in IEC 61508.Very often realization means to enhance a standard automation system with safety-security hardware and software features [2,3].
Hardware realization deals with the design of a hardware architecture including standard components such as a standard network access unit on a node, and the development of programmable and non-programmable hardware. For example, a node in an automation system uses a two channel architecture [3,14]. Or the input and output (I/O) unit requires additional circuits to test the I/O unit.
Safe-secure software to be integrated into software of an existing automation system is located above layer 7 (application layer) of the ISO/OSI reference model [14]. Examples are PROFIsafe or SafetyLon. Or an approach presented in [1] to secure LonWorks. Such an approach does not require to change the layers of the standard protocol stack and allows for interoperability aspects.
Hardware and software realization differs from traditional realization since very thorough quality requirements must be met. E.g., extensive and structured testing of hardware and software is necessary; or programming software with ANSI C is only allowed with a reduced set of commands. However, realization is based on standard hardware and software realization principles.

Use phase: The use phase is concerned with the installation, commissioning, operation, maintenance and disposal of the automation system. In opposition to the development phase that is unique for every entity of the system such as the node or gateway, the use phase activities relate to the whole automation system and must consider the overall interaction among all entities installed in a particular installation.

Overall installation and commissioning: In this activity the different entities are integrated into the automation system. Therefore, the safe-secure entities must be identified clearly in order to transfer the communication parameters to the designated nodes. The parameters like timing values are used to handle the safe-secure communication. Cryptographic keys applied to perform cryptographic operations like calculating a MAC must be distributed to the various entities.

Overall safety-security validation: In the field of safety (IEC 61508) the summary of safety requirements is considered to be the specification of intended use and system requirements. Hence, safety validation is the process of comparing system behavior with the safety requirements specification(s). In the context of security (IEC 15408), security objectives are a statement of indent. Seen from a more general point of view, safety-security validation is concerned with investigating if risks have been mitigated properly and the risk mitigation strategy is effective. E.g., validation means checking if integrity of the node is ensured constantly.

Overall operation, maintenance and repair: Operation covers all activities required to operate the automation system in a safe-secure way. Hence, the following issues should be addressed: how to test that the data transferred during commissioning was successfully stored on the designated entity; how to switch from commissioning to operation of the system; how to synchronize network time among nodes in case of applying a timestamp mechanism; how to update cryptographic keys; how to check availability of the field level network; and finally how to recover from network failures due to stochastic or systematic failures, or intentional attacks.
Maintenance and repair comprises activities such as how to gather diagnostic information in order to react to failures – with respect to security related failures gathering information is called a security audit. Another topic is the replacement of a safe-secure entity, or modification of communication parameters. Maintenance regarding reconfiguration of an automation system in general is a very challenging task. Just think of an airport with ten thousands of nodes. It is not acceptable to shut down the complete system when a node shall be replaced or configuration parameters shall be changed, just because safety and security ought not to be endangered. Sophisticated management of maintenance is necessary.

Decommissioning: It is the last stage of the safety-security lifecycle. There are three ways of understanding decommissioning: The whole system, an entity or a logical communication path between two entities can be decommissioned. Similar to maintenance a ordered decommissioning needs to be performed to maintain the safety-security of the system.

Conflict Resolution Approach

Safety and security more or less interact in every stage of the life cycle. Both show identities or conflicts. Of course, there are also areas where they do not interact. The requirements are considered to be independent. Identity means that safety and security strive for the same with equal or different effort. In opposition to identity, conflict indicates that safety and security pursue contradicting things.

Conflicts or identities between safety and security are always the result of conflicting or identical requirements or conflicting measures due to identical requirements. Consequently, to figure out and resolve conflicts between safety and security requirements, it is necessary to examine interdependencies. That is why activity conflict resolution is integrated into the development phase (Figure 2).

After activity 3 of the development phase safety requirements were specified that are part of the security environment. Additionally, a set of security requirements is available already considering safety requirements. What is still missing at this point is a cross-checking of both sets of requirements. Therefore, the conflict resolution approach in Figure 3 is applied. The result is a conflict-free set of requirements. Next, the measures implemented to satisfy the conflict-free requirements are checked. Cross-checking is only applied to these measures that are different although they result from the same requirement, stated once during safety and a second time during security requirement specification (Figure 4). After measures assessment has been performed, a threat-hazard and risk analysis is carried out to verify the correctness of the decisions made during conflict resolution and measure assessment. Especially, the conflict resolution policy is checked if it delivers the appropriate result with regard to the field of application of the automation system.

Requirement Level: Even though safety and security have the same major goal as mentioned in a previous section, they reduce risk to the goals because of different reasons. Put succinctly, safety is concerned with reducing risk to the system itself, whilst security strives for minimizing risk to information and resources referred to as assets resulting from malicious attacks. Accordingly, it is very likely that requirements how to minimize risk differ or even contradict each other. Hence, a methodology has to be specified that presents a clear and concise, and easy to handle way of conflict resolution. Such an approach is shown in Figure 3. It requires a specification of a conflict resolution first, a separation of requirements into two groups next to perform the conflict resolution itself afterwards.

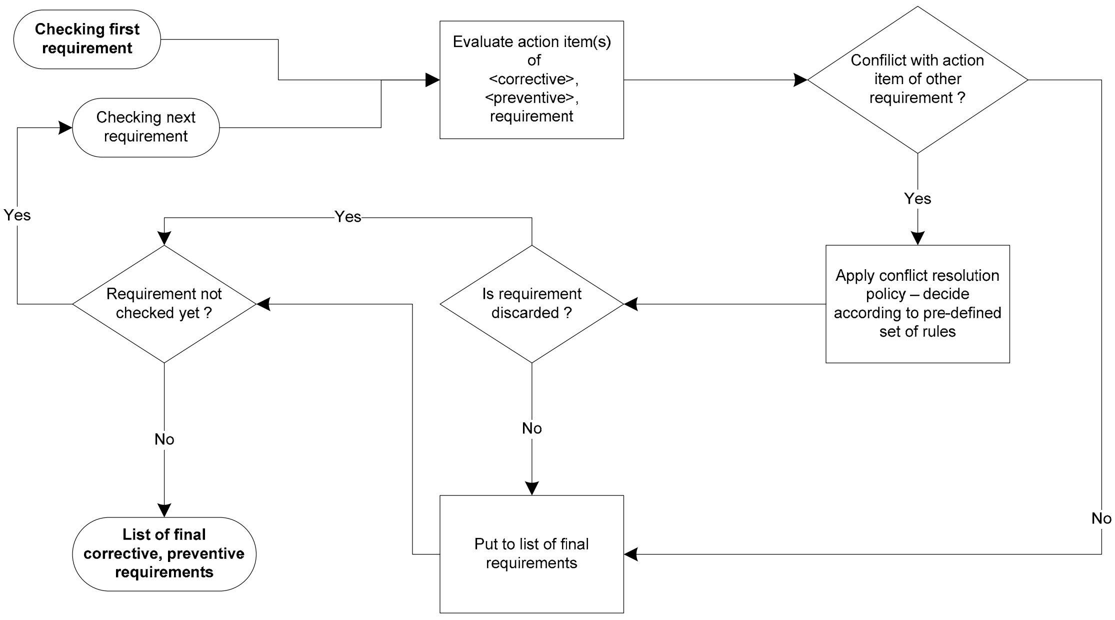

Checking first requirement
Evaluate action item(s) of <corrective>, <preventive>, requirement
Confilict with action item of other requirement ?
Checking next requirement
Yes
Yes
Yes
Requirement not checked yet ?
Is requirement discarded ?
Apply conflict resolution policy – decide according to pre-defined set of rules
No
No
No
List of final corrective, preventive requirements
Put to list of final requirements

Figure 3 — Conflict resolution at the requirement level

First, a conflict resolution policy must be specified. It is a set of one to many rules that enables the developer to allow for the particular point of view. The rules specify which requirement has to be preferred in a particular situation. A conflict resolution can be unique for the complete automation system, or a subsystem like a node, or even for the firmware of an entity in the system like a node. Within the paper the policy consists of two rules applicable to a subsystem, i.e. a node:

1. Prefer safety requirements to security requirements if security reduces safety.
2. Otherwise, use security requirement

Second, the requirements are categorized into three groups: a list of detective, and a list of preventive, corrective requirements. Detective requirements aim at detecting faults and attacks, e.g. 'Integrity of system data is detected'. Preventive requirements have the goal to prevent faults or attacks. In the safety domain such requirements are specified to avoid faults (fault avoidance), e.g. 'Software architecture is documented in accordance with a pre-defined layout'. Corrective requirements specify the corresponding response to a fault or an attack. For instance, 'In case of failure the node will return to a known state'.

Detective requirements result in actions that monitor the processes or the system during the different stages of the life cycle. They detect faults or attacks that have not been or could not have been prevented by measures derived from preventive requirements. For example, every message is checked regarding data integrity. A possible loss of integrity due to a stochastic failure or intentional attack is detected. How to react in case of loss of integrity is specified by a corrective requirement. Safety and security requirements in the detective category are identical, supporting or independent because they lead to actions that monitor and therefore do *not* modify the processes or system itself. By contrast, preventive and especially corrective requirements may influence processes or systems and are therefore subject to investigation in the conflict resolution approach.

Third, the conflict resolution itself is performed. Each requirement is evaluated regarding its action item (Figure 3). That is, the action and the reaction to a failure or attack is evaluated. The action item is checked against the other action items of corrective or preventive requirements. If there is no conflict, the requirement is considered to be a final safety-security requirement. Otherwise, the conflict resolution policy is applied and one of the requirements is discarded. In the end, a conflict-free set of requirements remains.

In the following an example is given to get an idea of the conflict resolution. Table 1 shows four corrective safety and security requirements, respectively. They are specified because of a failure or incident being detected by a detective function. The first failure is 'Hardware failure'. For instance, the online volatile memory test revealed a failure in a sector of the RAM of a controller on a node. From the safety point of view it is required that the controller switches to fail-safe state immediately. Since it is assumed that both controller must absolutely agree on every task that is executed (two channel architecture), fail-safe state of one results in fail-safe state of the complete node. Security requirement says that the first controller has to go to fail-secure state and the second one takes over. The conflict is solved by taking the safety and discarding the security requirement. In the case security reduces safety – rule one of the conflict resolution approach.

Table 1 — Corrective safety and security requirements

Failure/incident	**Safety requirement**	**Security requirement**
1. Hardware failure	Fail-safe state of node	Fail-secure state of safety chip
2. Integrity failure	Discard message	Discard message; after five successive incidents send message to network management device to signal attack.
3. Message lost	Fail-safe state of consumer	None
4. Disclosure of key	none	Stop communication until new key available

The second failure 'Integrity failure' results in similar safety and security requirements. Such a failure is detected due to CRC or MAC mismatches. Safety requirement is part of the security requirement. As security does not reduce safety, the security requirement is chosen – rule two of the conflict resolution approach. Failure 3 and incident 4 have no impact on security and safety, respectively. Consequently, the safety requirement as a reaction to 'Message lost' and the security requirement to 'Disclosure of key' is determined to be a final safety-security requirement.

Function Level: Conflict resolution at the function level or also called measure assessment is the second part of the conflict resolution approach. Measures are derived from the conflict-free set of safety-security requirements resulting from the first part of the conflict resolution. One of the many motivations to design a common approach is to benefit from synergies on applying measures. Consequently, safety and security measures and synergies gained are classified in three groups: (1) such that directly match (i.e. safety and security measure are equal), (2) such that are unique for safety and security, (3) and such that require different effort [13].

Measure assessment has to be only performed when safety and security measures are classified to be of group 3: different measures with different effort are derived from the same requirement. Only that group of measures exhibit conflicts regarding e.g., computational power, throughput, computation time, memory resources or application constraints. Measures of the other groups are either identical or unique and thus cannot be conflicted per definition. The conflict resolution on function level uses six factors as shown in Figure 4 to assess the safety and security measure. In the following an example of a measure assessment is given.

In a safe system the requirement 'Ensure integrity of data on a node or data in a message' is granted by means of a CRC. In a secure system similar measures are used. Instead of the CRC a cryptographic message authentication code (MAC) is used that cannot be recalculated without the knowledge of the appropriate key (see Figure 1). According to the node address and in case of a proven authenticity of the message, data can be read or written.

To gain a synergy, a proper solution is to replace the CRC by the MAC since the security measure also withstands intentional attacks to the integrity of data. Whether this is a valid and effective replacement is evaluated by the measure assessment. First, safety integrity must not be jeopardized, i.e. the same safety integrity level (SIL) must be reached as it was before the replacement of the safety measure. In other words, a MAC must be selected that grants the same level of integrity than the non-secure CRC does. Second, according to the evaluation assurance level (EAL) a minimum strength of function is specified and has to be proven. Put another way, the MAC has to be chosen in a way so that it cannot be defeated. Third, software and hardware environment has to be considered. Selecting a cryptographic algorithm implemented in hardware, results in less computational time than implementing the same algorithm in software. Furthermore, memory resources of embedded devices are low compared to PCs and have also to be considered. Finally, performance of measures and the impact on field of application are examined. Referring to [13], generating and verifying a 8 byte MAC on a smartcard takes about 50 ms of time whereas the process of calculating a CRC lasts about 300-500 µs. Consequently, ensuring integrity with a MAC excludes applications like emergency push button since an overall reaction time, i.e. time from pressing the button on Node A and stopping a machine connected to Node B, of less than 150 ms is required [14].

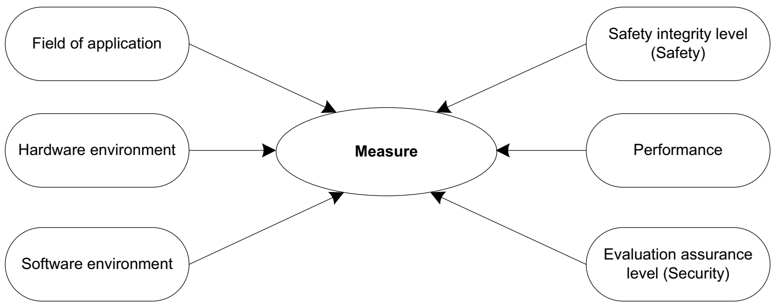


Figure 4 — Conflict resolution at the function level

Conclusion

The possibilities to gain synergies by taking an common approach towards safety and security in building automation is given in many areas: on non-functional measure level like verification and validation tools or risk analysis techniques. However, it is not well-known that synergies can be gained at the requirement level or at functional measure level (CRC vs. MAC) as presented in the paper. Gaining synergies is possible since safety and security strive for some identical goals. This fact is often honored, yet little effort to combine the fields is given because applications are thought to be either safety or security critical.

In the paper a safety-security life cycle model that attempts to integrate safety *and* security in an automation system is presented. The life cycle model (Figure 2) allows for safety and security topics throughout the whole life of a system to ensure a high level of confidence in the safety-security quality. In particular, the paper focuses on how to resolve conflicts between safety and security requirements and measures. Such conflicts are likely because both domains have similar goals, but they are implemented differently. As a consequence, a two step conflict resolution framework is presented, resolving conflicts at the requirement and at the function level. Such a life cycle model and its included conflict resolution are the basis of combining formerly separated networks for safety, e.g., fire alarm system, and for security, e.g., access control, and operation, e.g. heating, ventilation and air condition.

References

1. C. Schwaiger, A. Treytl. Smart Card Based Security for Fieldbus Systems. In *Proceedings of 9th IEEE Conference on Emerging Technologies and Factory Automation*, Volume 1, pp. 398-406, 2003.
2. A. Treytl, T. Sauter, C. Schwaiger. Security Measures in Automation Systems - a Practice-Oriented Approach. In *Proceedings of 10th IEEE International Conference on Emerging Technologies and Factory Automation*, Volume 2, pp. 847-855, 2005
3. T. Novak, T. Tamandl. Architecture of a Safe Node for a Fieldbus system. In *Proceedings of the 5th IEEE International Conference on Industrial Informatics*, Vol. 1, pp. 101-106, 2007.

4. M. Naedele. Standardizing industrial IT security - a first look at the IEC approach. In *Proceedings of Emerging Technologies and Factory Automation ETFA 2005*, Volume 2, pp. 19-22, 2005.
5. U. Baumgarten, C. Eckert. Mobil und trotzdem sicher?. *it+ti* 5/2001, pp. 254-263, Oldenbourg Verlag, 2001.
6. Institute of Electrical and Electronics Engineers Power Engineering Society. *IEEE Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations*. IEEE Std 7-4.3.2, 2003.
7. W.F. Bates. Safety-related system design in power system control and management. In *Proceedings of the 4th International Conference on Power System Control and Management*, pp. 15-20, 1996.
8. International Electrotechnical Commission. *IEC 61508 – Functional safety of electric/electronic/programmable electronic safety-related systems*. IEC, 1998.
9. D. J. Smith, K. G. L. Simpson. *Functional Safety – A straightforward guide to applying IEC 61508 and related standards*, Elsevier Butterworth-Heinemann, Oxford, 2nd edition, 2004.
10. P. Wratil, M. Kieviet. *Sicherheitstechnik für Komponenten und Systeme*. Hüthig Verlag, Heidelberg, 2007.
11. G. McCraw. *Software Security – Building Security In*. Addison-Wesley, Boston, 2006.
12. International Electrotechnical Commission. *IEC 15408 – Information technology – Security technique – Evaluation criteria for IT security*. IEC, 2nd edition, 2005.
13. T. Novak, A. Treytl, P. Palensky. Common Approach to Functional Safety and System Security in Building Automation and Control Systems. In *Proceedings of the 12th IEEE International Conference on Emerging Technologies and Factory Automation*, pp. 1141-1148, 2007.
14. D. Reinert, M. Schaefer (Publisher). *Sichere Bussysteme in der Automation*. Hüthig Verlag, Heidelberg, 2001.
15. A. Burns, J. McDermid, J. Dobson. On the meaning of Safety and Security. *The Computer Journal*, Vol. 35, No. 1, pp. 3-15, 1992.

Biography

Thomas Novak, Ph.D., Research Assistant, Institute of Computer Technology, Vienna University of Technology, Vienna, Austria. Telephone +43-1-58801 38427, e-mail: novakt@ict.tuwien.ac.at.

Dr. Thomas Novak was born in Vienna, Austria and works in the field of functional safety and security in building automation for more than three years. He is project leader at the Institute of Computer Technology, Vienna University of Technology. He is chair member of the Austrian Smart Card Association (ASA) and expert in the European Standardization CEN, Technical Committee (TC) 247, Working Group (WG) 4 where he is mainly involved in setting up a standard for functional safety and security in building automation. Additionally, he is member of the IEEE IES technical committee building automation, control and management (TC BACM).

Albert Treytl, Researcher, Research Unit for Integrated Sensor Systems, Austrian Academy of Sciences, Wiener Neustadt, Austria, Telephone +43-2622-23420-0, e-mail: albert.treytl@oeaw.ac.at.

Albert Treytl worked as research assistant at the Vienna University of Technology in the area of energy management systems, field bus communication systems and as leader of the security activities. The focus of his research lies on security of automation networks such as field busses and wireless ad-hoc networks. The topics of his research projects included web-based interfacing of energy management systems, electronic payment, smart card development and research, and the planning of security architectures. Currently he is responsible for the security and vertical integration aspects at the research unit for integrated sensor systems. Aside this he engages himself in various technical committees (IEEE, CEN TC 247 WG4, IEEE1588 standardization) and scientific conferences.

Andreas Gerstinger, Research Assistant, Institute of Computer Technology, Vienna University of Technology, Vienna, Austria. Telephone +43-1-58801 38457, e-mail: gerstinger@ict.tuwien.ac.at.

Andreas Gerstinger is a research assistant at the Institute of Computer Technology, which is part of the University of Technology in Vienna, Austria. He works in a project which is sponsored by two companies which both build systems for safety-critical domains. In this function, he is on the interface between academia and industry. Before that, he has worked for six years as a technical lead for safety and software quality at the Frequentis corporation. He holds a master's degree in computer science and telecommunication.