COLLABORATIVE SECURITY ASSESSMENTS IN EMBEDDED SYSTEMS DEVELOPMENT

The ESSAF Framework for Structured Qualitative Analysis

Friedrich Köster, Michael Klaas, Hanh Quyen Nguyen, Walter Brenner
Institute of Information Management, University of St. Gallen, Mueller-Friedberg-Str. 8, 9000 St. Gallen, Switzerland

Markus Brändle, Sebastian Obermeier
ABB Research, Segelhofstr. 1K, Post Box, 5405 Baden, Switzerland

Keywords: Collaborative security assessment, ESSAF framework, Embedded systems security, Security knowledge management, Threat modeling.

Abstract: The standardization of network protocols and software components in embedded systems development has introduced security threats that have been common before in e-commerce and office systems into the domain of critical infrastructures. The ESSAF framework presented in this paper lays the ground for collaborative, structured security assessments during the design and development phase of these systems. Its three phases system modeling, security modeling and mitigation planning guide software developers in the independent assessment of their product's security, minimizing the burden on security experts in the collection of security relevant data.

1 INTRODUCTION

The use of networked embedded systems for the monitoring and control of critical infrastructures, such as energy, telecommunication or transportation networks and factory automation systems, has introduced similar IT security problems into these domains that have been existing in e-commerce and office networks for a long time (Byres and Lowe 2004). The increasing use of commercial off-the-shelf (COTS) software and standardized protocols such as TCP/IP in the development of these embedded systems increases both the number of vulnerabilities – due to better accessibility of the systems – and threats – due to more widespread knowledge about the vulnerabilities (Igure, Laughter et al. 2006).

Systematically assessing the security requirements and issues of these systems during the development phase is a crucial task in the design of any embedded system for critical infrastructures. Whenever a security assessment of such systems must be conducted, a multitude of stakeholders from different disciplines and with a diverse background should work together. Each of the stakeholders involved in building and assessing such systems has different pieces of information and different expertise, which can only contribute to a comprehensive evaluation of a system's security if it is systematically collected and documented in a well-defined way.

1.1 Contributions

In this paper, we describe the ESSAF framework (Embedded System Security Assessment Framework), which provides a method and a software tool for the collaborative evaluation and documentation of embedded systems during their design and development phase. The framework includes a software tool currently under development, which supports the assessment and knowledge sharing process. The framework integrates techniques for system modelling, security requirements documentation, threat modelling, risk and mitigation management with collaboration and knowledge management techniques together with a consistent data model for the structured analysis of embedded systems. The method and tool enable

Köster F., Klaas M., Quyen Nguyen H., Brenner W., Braendle M. and Obermeier S. (2009).
COLLABORATIVE SECURITY ASSESSMENTS IN EMBEDDED SYSTEMS DEVELOPMENT - The ESSAF Framework for Structured Qualitative Analysis.
In *Proceedings of the International Conference on Security and Cryptography*, pages 305-312
DOI: 10.5220/0002189903050312

system experts without specific security knowledge to participate in the evaluation process. They help to make security assessments more efficient than current "offline" or "brainstorming" based methods while enabling the integration of ongoing security evaluations into the development process.

1.2 Related Work

Several methods for the modeling of threats and vulnerabilities or the identification and assessment of IT security risks exist. Especially the field of "probabilistic" risk management approaches, which focus on the formulation of scenarios and their evaluation in terms of probability of occurrence and (monetary) impact, is now well described (Ralston, Graham et al. 2007). Sometimes, the monetary impact quantification is replaced or combined with other units such as loss of life or severity of injury (Tolbert 2005). National or international standards such as (Standards Australia & Standards New Zealand (SA/SNZ) 2000) and (ISO/IEC 2005) describe the organization of risk management processes that will also consider information security risks. They cover multiple phases of a risk management process, with a focus on the organizational aspects.

In order to effectively solve the given assessment task, the abstraction level of the method needs to be considered. The well-known OCTAVE method (Alberts and Dorofee 2001), (Alberts, Dorofee et al. 2003) has a focus on a strategic, high-level description of the risk assessment process, but leaves the choice of techniques to the users. This requires constant involvement of a specialist that can choose suitable techniques for data collection. Collaboration is often organized in the form of physical meetings with all stakeholders together with a security expert, as in the CORAS method (Vraalsen, den Braber et al. 2004).

A security assessment method that takes into account the technical implementation of systems has been described by Microsoft researchers in (Swiderski and Snyder 2004) and (Howard and Lipner 2006). This method uses information about damage potential and affected users in order to rate risks, which is not generally known in the development scenarios which are targeted by this method.

Aspects of collaboration in security assessments for IT systems without regard for a specific method have been covered by (Steffan and Schumacher 2005) for the area of attack modeling. The concept of using a knowledge base in security risk management has been described by (Kailay 1995). Organizational and practical challenges in connection with IT security for critical control systems are described in (Naedele 2007).

1.3 Roadmap

We start in section 0 with a description of the ESSAF method. Section 0 describes the processes for collaboration. The description of knowledge sharing mechanisms can be found in section 0. Our findings are summarized in section 0.

2 METHOD OUTLINE

The main focus of the ESSAF method is to document what to protect (security objectives of assets) how (rationales, security measures) from which threats. The assessment process is divided into the three phases System Modeling, Security Modeling and Mitigation Planning, which are sketched in the process model shown in Figure 1.

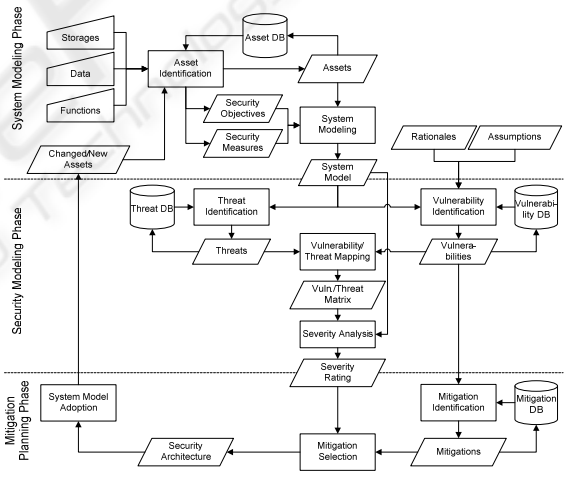

Figure 1: Iterative process Model of the three phases to 1: collect information about the system, its security objectives and security measures, 2: document and asses the system's security and 3: plan necessary changes to improve the system's security architecture.

Even though the three phases are shown in consecutive order, it is not necessary to fully complete one phase before entering the next. For example, a vulnerability can be entered for an asset before the whole system model is completed. The idea is that as more information is entered into an assessment, the necessary cross-references between assets, security measures, rationales and threats will prompt for an iterative completion of the

Table 1: Some security objectives only apply to data assets, whereas others are useful to describe security needs of functions. Storages, for example, inherit the security objectives of the data they hold, but they can have their own availability requirement.

	Authenticity	Access Control	Auditability	Confidentiality	Integrity	Availability
Function	x	x	x			x
Data				x	x	
Storage				x	x	x

information. Also, the overall process is designed to be carried out concomitantly to the design and development activities for the modeled system.

2.1 System Modeling

The system modeling phase starts by documenting the system's assets and their relationships in a data flow diagram: Data, functions and storage entities are related to each other by data flows that connect them. A suitable level of abstraction should be chosen in order to keep the system model understandable to participants that are not involved in the actual implementation of the system (Schuette and Rotthowe 2004); not every implemented software function needs to become a function in the data flow diagram, and multiple data elements can be summarized as one entity if they are of the same type and have the same security objectives (e.g. different control data can be modeled as one entity).

Assets are annotated with *security objectives*, which represent the required security level of a system. In a model instance, each security objective defines a protection goal for a specific asset. The following six commonly used security objectives according to (Dzung, Naedele et al. 2005) are pre-defined, but more specialized security objectives can be added if needed (e.g. non-repudiability, plausible deniability, third-party protection (Ma 2004)):

Authenticity: the identity of a user (e.g. an operator or another process) must be verifiable

Access Control: access to resources is limited to authorized users based on rules; the authorization requires prior authentication

Auditability: it is possible to identify the user that performed an operation

Confidentiality: data may not be disclosed to unauthorized users

Integrity: data must be protected during transmission and storage against unauthorized modification or destruction

Availability: a resource must be accessible or usable upon demand by a legitimate entity or process

Table 1 shows the security objectives that are applicable to each type of asset. For all applicable security objectives for each system component, ESSAF requires the security assessors to provide a reasoning if the security objective is deemed to be irrelevant.

When the required security level has been described by all assets' security objectives, the current security level of a system needs to be documented in rationales. A rationale describes how a security objective is achieved (or not achieved, or partially achieved) for a given asset. A rationale may use a security measure, which is security functionality provided by one of the system's components. An example of a rationale using a security measure would be: "User credentials are encrypted by the web server's SSL function". If a rationale contains an assumption, the assumption must be explicitly formulated (techniques for uncovering hidden assumptions are described in (Bishop and Armstrong 2005)). In this case, one assumption is that "the web server's SSL function will always be used if user credentials are transmitted over a network". The list of all assumptions can later be used by other stakeholders who are concerned with the deployment or use of the system in order to make sure that all documented assumptions are met in the implementation.

2.2 Security Modeling

In order to check whether the current security architecture can uphold against the intended security requirements, the documented security objectives need to be mapped against the currently implemented security measures in a matrix. By reasoning about the sufficiency of each security measure, gaps can be identified where the currently implemented measures are not enough in order to ensure the security objectives. There may not be security measures in place for all the security objectives of all assets. In this case, the reasoning and assumptions behind the lack of security measures has to be documented in the rationale.

Vulnerabilities

The question whether the implementation of a security measure or other functionality is correct and sufficient in order to uphold a security objective is not in the scope of this method, but needs to be assessed by (security) experts, possibly with the help of other tools such as port scanners, static code validation and external vulnerability databases (Viega, Bloch et al. 2000). It is possible that there is a security measure in place, but the technical implementation is not strong enough or can be circumvented in some way. It is not intended to automatically detect this kind of flaws, but known vulnerabilities should be stored in the knowledge base and can then be checked off each time a security measure is introduced.

Another starting point for the discovery of vulnerabilities is the analysis of all assumptions. Whenever an assumption from a rationale does not hold true, this will very likely introduce a new vulnerability into the system.

A third possibility to support vulnerability discovery is a rule based approach. The main idea behind rule based vulnerability discovery is to capture security expert knowledge into a set of rules that can be automatically (or in a check-list approach) applied to a system model in order to detect inconsistencies or "weak spots" that warrant a more careful analysis by a human expert in order to decide whether it is a vulnerability or not. The rules can be formulated in predicate logic. An example for a predicate logic rule that would point out that some confidential data has not been explicitly modeled may be: "Exists *Function SSL Server* And Not Exists *Data SSL Certificate"*. This approach is applicable for system elements which are classified hierarchically (cf. section 0), so that the semantics of the elements are known.

Threats

A threat is described in terms of a possible attack scenario that includes

- the targeted assets and the security objectives that are endangered as well as the possible motivation for this compromise,
- the associated assets that also play a role in this attack,
- the vulnerabilities that could be exploited in order to realize this goal and
- an approximated classification of the costs or complexity of the attack (low, medium or high).

The complexity rating can be made according to the criteria that are described under "Access Complexity (AC)" in (Mell, Scarfone et al. 2007).

Severity Rating

A system developer of a sufficiently generic embedded system cannot quantify risk very well by means of probability and impact estimates due to limited knowledge about later usages of the system. Nonetheless, vulnerabilities need to be prioritized in order to make informed decisions about mitigation measures. Much of the information is already contained in the textual descriptions of the assets' rationales and in the system model itself. In order to increase the manageability of the found vulnerabilities, a simple severity ranking scheme is proposed based on four factors:

1. number of endangered security objectives of assets,
2. importance of the endangered security objectives,
3. number of threats that could exploit the vulnerability and
4. exploitability (costs) of the threats for the vulnerability.

Table 2: Derivation of the impact level for a vulnerability.

Level	Description
High	at least three affected security objectives of "default" importance *or* at least one affected "critical" security objective
Medium	at least two affected security objectives of "default" importance
Low	one affected security objective of "default" importance *plus* any number of affected security objectives of "low" importance

Table 3: Derivation of the exploitability level for a vulnerability.

Level	Description
High	at least two threats that exploit the vulnerability *or* at least one threat with low attack costs that exploits the vulnerability
Medium	at least one threat with medium or high attack cost that exploits the vulnerability
Low	no known threat that exploits the vulnerability

In a first step, factors 1. and 2. can be combined into an "impact" figure for each vulnerability (cf. Table 2), clearly focusing on the technical impact, not the business impact. Factors 3. and 4. can be combined into an "exploitability" figure (cf. Table

3) with the help of a vulnerability-threat-matrix that indicates which threats (if any) exploit what vulnerabilities. Each vulnerability can then be mapped in a *severity matrix* that has the impact rating on the x-axis and the exploitability rating on the y-axis (cf. Figure 2). The reasoning behind the exploitability rating acknowledges that even a vulnerability for which no threat is known can be very critical, as there is no technique which could assure that all threats have been documented. On the other hand, it does not ignore the threat information and gives a "best estimate" for the likelihood of attacks based on the known information.

2.3 Mitigation Planning

Mitigations address the found vulnerabilities in order to attenuate their effect, to provide additional countermeasures or to completely eliminate them. All possible mitigations are described with the following information:

- security measures that will be introduced by the mitigation,
- vulnerabilities that the mitigation is supposed to fix,
- approximate cost of mitigation measure in person days and/or monetary value and
- possible negative effects on the system (feasibility).

As development resources are always limited, planning is needed in order to find a set of mitigation measures which will lead to the greatest improvement to the system's security with the available resources under the constraints of technical feasibility and possible side effects. The basic steps of this process are as follows:

1. Identify possible mitigation measures for the found vulnerabilities; relate all vulnerabilities and mitigation measures.
2. Assess the approximate costs and mitigating effects for all measures; plan what new assets and security measures will be introduced by the mitigations and what the overall effect will be on the whole system.
3. Select a set of mitigations for implementation (by expert reasoning).
4. Go through the process for the new system model and adjust the system model and the vulnerability, threat and risk analysis accordingly.

The implementation of new mitigations will introduce new assets, security measures and security objectives into the system model, and most likely, even new threats and vulnerabilities will arise from the changes. This makes it imperative to go through a new iteration of the security assessment process once the new mitigations have been selected. The process will only stop once it is decided that no more mitigations are reasonable or feasible.

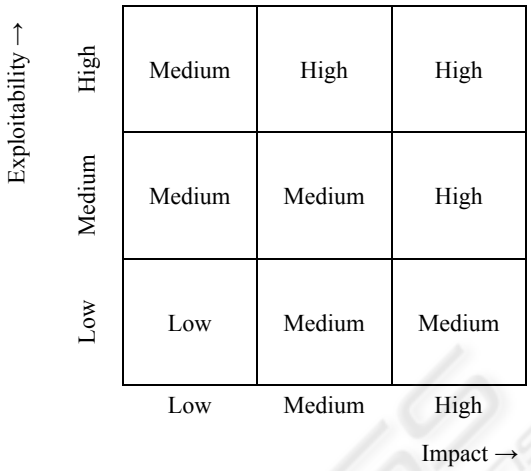

Figure 2: Severity matrix for vulnerabilities.

3 COLLABORATIVE SECURITY ASSESSMENTS

We divide the collaboration features into two independent processes: 1) The storage, sharing and collaborative creation of single assessments (described in this section) and 2) the creation, editing and sharing of an inter-divisional common knowledge base of findings (see section 0). During an assessment, it is important to capture the knowledge of as many stakeholders as possible. The role model described in section 0 prescribes which roles need to be filled as a minimum prerequisite for security assessments with validation by cross-checking of the entered information.

3.1 Role Model

Practical experience shows that it is hard to start a security assessment "on a blank page". In order to get started easily, the method begins by documenting the functions, data and storages of a system, which are well known to developers and system architects. Since a system architect is used to creating abstract models of systems and has the best overview of the system, he is tasked with the creation of an initial system model with the most important components. The system architect can

Table 4: Roles in the creation of a security assessment.

Role	Tasks and responsibilities
System Architect	Creates a first, complete overview of the involved system components and their relations. Documents rationales and existing security measures, identifies vulnerabilities. Provides feasibility judgments about possible mitigations.
System Developer	Provides specific input about the components in parts of the system. Identifies vulnerabilities.
Product Manager	Assigns security objectives to the system components. Checks plausibility and completeness of the system model. Describes and rates threats. Provides cost figures for mitigations.
Security Expert	Checks the plausibility and completeness of found vulnerabilities and threats. Proposes mitigation measures.
Business Manager	Checks the overall plausibility of the model and its assumptions. Approves versions of the assessment. Decides about mitigation measures.

then delegate the detailed definition of parts of the system to system developers that are familiar with this part of the system. A product manager then defines what security objectives are needed for the system's assets. For each security objective of each asset, the system architect (possibly again with the help of system developers) needs to provide a rationale that states how the security objective is achieved, also modeling the security measures involved. A crucial part is the identification of vulnerabilities, which is mainly the task of system architect and developers, but can be supported by the security expert.

The assumption is that the role of the security expert is the hardest to fill, and the security expert will have the highest time constraints. One goal of the method thus is to minimize the effort for the security expert not directly related to the discovery and treatment of security issues. Therefore, the identification of vulnerabilities will also be supported by a central knowledge base (see section 0). The security expert needs to judge whether vulnerabilities or threats have been overlooked, and he can propose mitigation measures such as suitable security measures or changes to the existing architecture or functions. When the system has reached a stable development state, the business manager is responsible to approve the assessment when he has the impression that it has been carried out diligently and reflects all major risks. The business manager also decides on mitigation measures based on the information about effects and costs provided by the system architect and the product manager. A summary of all roles that exist for a security assessment is provided in Table 4.

3.2 Validation by Structure

An important effect of collaborative modeling is that inconsistencies and different views about the "appropriate" representation of the system structure or security architecture often lead to the discovery of vulnerabilities. For example, when the system architect has to provide a rationale for an asset's security objectives, he may use an assumption or a security measure. The system developers may then discover if they violate one of those assumptions, or if a security measure is used in a way that it was not intended for (e.g. using a checksum function to check for data integrity, which is not correct in a security context).

Furthermore, the structure of the system model itself can help to uncover vulnerabilities. For example, a security measure such as "input validation" has to be attributed to a function in the system model that provides it. It is well conceivable that a system architect creates this security measure, but finds no function where to assign it. In the process of finding a developer which is responsible for this functionality, it may turn out that no suitable function can be found that provides input validation.

4 KNOWLEDGE SHARING

An additional support for security assessment collaboration is a common knowledge base that holds assessment objects and their related security information (e.g. security objectives, threats, and vulnerabilities) of previous assessments. The knowledge base fulfills three main goals: 1) To make the creation of new assessments more efficient by re-using objects from the knowledge base, 2) to validate individual assessments and check their plausibility against information from the knowledge base (Steffan and Schumacher 2005) and 3) to enable inter-divisional communication about security issues that pertain to objects from the knowledge base which have been used in more than one assessment.

4.1 Knowledge Base Content

When deciding on what to store in the knowledge base for later re-use, it is important to distinguish between "negative" and "positive" security information. "Negative" information informs about required security levels, which may or may not be accomplished (e.g. security objectives, vulnerabilities, threats). "Positive" security information explains how the security levels are assured (e.g. rationales, assumptions, security measures); if missing, it will raise a "red flag" that prompts for action – either further documentation of the measures in place, or mitigation measures that remove an assets' vulnerability. If "positive" security information is delivered from the knowledge base, this would carry the risk that rationales and security measures are adopted without checking if they are applicable in the new context. Because the reasoning behind the security architecture needs to be done by those responsible for a system, no rationales, assumptions or reasoning about why a security objective is *not* relevant to an asset can be stored in the knowledge base.

4.2 Administration and Organization of the Knowledge Base

Good data quality and sufficient retrieval methods determine the success of the central knowledge base. In order to support these goals, the knowledge base can only be filled by a dedicated knowledge base administrator. Once the security assessments of different teams are uploaded to a central server, the administrator evaluates them for system and "negative" security information that can be entered into the knowledge base. The editor can also group entities and assign a common name to this group of entities (e.g. a set of functions, storages and data flows which have a common purpose and usually exist together). When components are taken from an individual assessment, their name may be changed by the editor in order to adhere to a common naming scheme in the knowledge base. The components will still be identifiable by their assessment and object IDs which are not changeable by the editor. The knowledge base is stored and edited centrally on a server; the local client applications regularly download the current version of the knowledge base (or the changes since the last download, respectively). New information cannot be entered directly by clients into the knowledge base, but needs to be extracted from assessments stored on the server. The process of storing, editing and sharing knowledge is outlined in Figure 3.

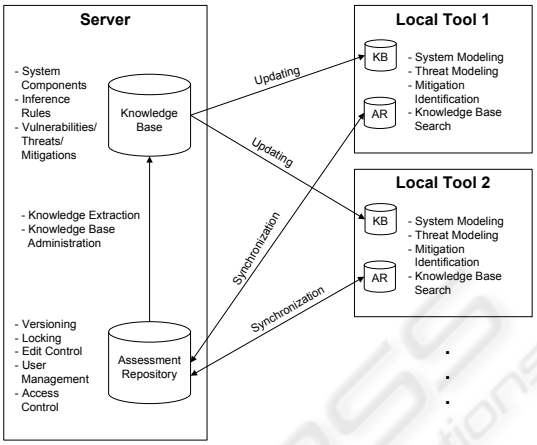

Figure 3: Collaboration between different users through a central server.

In order to identify and retrieve the objects from the knowledge base, it is essential to be able to classify them according to a common scheme. This classification is achieved by assigning tags to each system component. The tags can then be organized in a hierarchy for each entity class (i.e. one hierarchy for all functions, one for all data elements, etc.).

This improves the organization of knowledge in two cases: When a new object is created by an assessment user and is then marked with a tag that already exists in the knowledge base, it is possible to automatically propose using an existing system component from the knowledge base which may not have been considered for re-use when entering the new entity in the assessment. And for administering the knowledge base, this hierarchy can speed up the process of finding duplicate entities in order to decide if some new information should be entered into the knowledge base or not. Security information such as vulnerabilities and threats are assigned to the system components that they can affect, and can be found by searching for their associated system components.

4.3 Update Notification

When a system component from the knowledge base is re-used in another assessment, it "remembers" its ID, just as components that are entered into the knowledge base also retain their original assessment and object IDs. In this way, if new "negative" security information such as security objectives, vulnerabilities or threats are discovered in one of the assessments where the component is used, all other

assessment projects can be notified of this new information after the next download of the central knowledge base. The assessment teams can mark the new information as “irrelevant” or “mitigated”, but they have to provide a reasoning why the vulnerability, threat, etc. is not relevant in their system.

5 CONCLUSIONS

The ESSAF framework enables the collaborative, structured documentation of an embedded system’s architecture, its components and their security objectives and security measures as a basis for a systematic analysis of vulnerabilities and threats to the system. This allows for informed decisions about a mitigation strategy for the identified vulnerabilities. Because assumptions and the reasoning behind evaluations are documented, the traceability of the results is ensured.

The analysis can be carried out by system designers and developers, who do not need to be security experts, as part of their daily work. The assessment information can evolve gradually as more information is provided by different stakeholders. The resulting documentation serves as the basis for further analysis by a security expert, who can save effort in data collection. The data structure also allows for the use of a knowledge base that can assist in the identification of inconsistencies and possible vulnerabilities.

No information about concrete use cases is needed for the analysis and rating of vulnerabilities, especially no probability or monetary impact figures have to be indicated. This supports the use of the method during the design and development of embedded systems that may be deployed in very diverse settings later. The supporting software tool enables the structuring of information.

ACKNOWLEDGEMENTS

This work was supported by the Swiss Confederation’s innovation promotion agency CTI.

REFERENCES

Alberts, C. and A. Dorofee (2001). OCTAVE Method Implementation Guide Version 2.0. Pittsburgh, PA, USA.

Alberts, C., A. Dorofee, et al. (2003). "Introduction to the OCTAVE Approach." Retrieved 2007-03-05, from http://www.cert.org/octave/approach_intro.pdf.

Bishop, M. and H. Armstrong (2005). Uncovering Assumptions in Information Security. WISE4 Forth World Conference "Information Security Education". Moscow, Russia, Moscow Engineering Physics Institute (State University): 223-231.

Byres, E. and J. Lowe (2004). 'The Myths and Facts behind Cyber Security Risks for Industrial Control Systems'. VDE Congress. Berlin.

Dzung, D., M. Naedele, et al. (2005). "Security for Industrial Communication Systems," *Proceedings of the IEEE*, 93 (6): 1152-1177.

Howard, M. and S. Lipner (2006). *The Security Development Lifecycle*, Microsoft Press, Redmond, WA.

Igure, V. M., S. A. Laughter, et al. (2006). "Security issues in SCADA networks," *Computers & Security*, 25 (7): 498-506.

ISO/IEC (2005). 27002:2005 Information Technology. Code of Practice for Information Security Management. Geneva, Switzerland, International Organization for Standardization (ISO).

Kailay, M. P. J., Peter (1995). "RAMeX: a prototype expert system for computer security risk analysis and management," Computers & Security, 14 (5): 449-463.

Ma, Q. (2004). A study on information security objectives and practices. Department of Management. Illinois, Southern Illinois University.

Mell, P., K. Scarfone, et al. (2007). CVSS - A Complete Guide to the Common Vulnerability Scoring System, Version 2.0.

Naedele, M. (2007). Addressing IT Security for Critical Control Systems. 40th Hawaii Int. Conf. on System Sciences (HICSS-40). Hawaii.

Ralston, P. A., J. H. Graham, et al. (2007). "Cyber security risk assessment for SCADA and DCS networks," *ISA Transactions*, 46 (4): 583-594.

Schuette, R. and T. Rotthowe (2004). "The Guidelines of Modeling - An Approach to Enhance the Quality in Information Models," *Lecture Notes in Computer Science*, 1507: 240-254.

Standards Australia & Standards New Zealand (SA/SNZ) (2000). AS/NZS 7799.2:2000 Information Security Management. Homebush, Australia; Wellington, NZ, Standards Australia & Standards New Zealand.

Steffan, J. and M. Schumacher (2005). 'Collaborative Attack Modeling'. ACM Symposium on Applied Computing. 2005-03-13.

Swiderski, F. and W. Snyder (2004). *Threat Modeling*, Microsoft Press, Redmond, WA.

Tolbert, G. D. (2005). "Residual Risk Reduction," *Professional Safety*, 50 (11): 25-33.

Viega, J., J. T. Bloch, et al. (2000). 'ITS4: A static vulnerability scanner for C and C++ code'. 16th Annual Computer Security Applications Conference (ACSAC'00). New Orleans, Louisiana.

Vraalsen, F., F. den Braber, et al. (2004). The CORAS tool-supported methodology for UML-based security analysis. Trondheim, Norway, SINTEF.