Implications of the Operational Environmental on Software Security Requirements Engineering

Christian Schmitt[1] and Peter Liggesmeyer[2,3]

[1]IT Security Consultant Siemens AG, Siemens Corporate Technology,
Otto-Hahn-Ring 6, 81739 Munich, Germany
[2]Research Group Software Engineering, University of Kaiserslautern,
67653 Kaiserslautern, Germany
[3]Fraunhofer Institute for Experimental Software Engineering, 67663 Kaiserslautern, Germany

Abstract. After presenting an overview about the most commonly referred reasons and issues for bad practice in software security requirements engineering, this paper introduces a security interdependency model, illustrating the implications between software and its physical, technical and organizational environment. The model is described in detail and the mutual implication and interdependencies between software security (requirements) and the operational environment are explained, enhanced with illustrative examples. Conclusions and further research perspectives with respect to security requirements engineering, and security in general are drawn.

1 Introduction

Poor requirements engineering and management is one of the most relevant reasons for software project cancellations [1]. Moreover, most vulnerabilities and weaknesses in software systems originate from inadequate or incomplete security requirements [2].

Although the actual financial damage is hard to quantify, security flaws cause a negative financial impact of several billion USD a year [3]. The costs of correcting an error increase rapidly with each phase of the development life cycle that the correction is delayed [4]. In other words, errors made during the requirements engineering phase have a high negative impact on the overall success as well as on time, budget and security of software development project. From a security perspective these perceptions motivate that security considerations should be done in the earlier phases of the development lifecycle. In recent years the former paradigm to 'penetrate and patch' was replaced by security software development at the design and coding stage [5], however it did not yet reach the requirements engineering phase in many cases. Practical experience and a recent survey [6] indicate that security requirements are seldom explicitly elicited and documented as part of the requirements engineering phase, but instead are mostly considered during the implementation phase.

In literature on security requirements engineering it is often stated that the application of security requirements engineering methods and techniques in projects and the maturity of security requirements specifications are mostly poor (e.g. [7]) or do not

Schmitt C. and Liggesmeyer P..
Implications of the Operational Environmental on Software Security Requirements Engineering.
DOI: 10.5220/0004966400630074
In *Proceedings of the 11th International Workshop on Security in Information Systems* (WOSIS-2014), pages 63-74
ISBN: 978-989-758-031-4

exist. “Most of the work for traditional requirements engineering fails to explicitly consider security” [2]. Mead states in [8] “When security requirements are considered at all during the system life cycle, they tend to be general lists of security features such as password protection, firewalls, virus detection tools, and the like”.

Although much work has been done in the recent years to improve the overall maturity of the security requirements engineering domain, still a lot of uncertainties and inconsistencies exist and the practical application of existing methods and the maturity of security requirements specifications needs improvement. Security requirements engineering is still a challenging task, requiring profound security and security requirements engineering method knowledge.

It is the objective of this paper to examine the interdependencies between software and the environment, as one important component of security requirements engineering. In section 2 we present a summary of frequently mentioned issues leading to poor security requirements engineering, which serve as basis to derive and explain the motivation for our focus topic about the environmental implications on security requirements engineering. Section 3 provides a security dependency model how to visualize and describe the interdependencies between software and its environment, and explains the implications (i.e. constraints, influences and assumptions) on security requirements engineering. For a better understanding, the model is explained using two examples. In section 4 we summarize, draw conclusions and reflect these with the associated commonly referred issues in security requirements engineering (as given in section 2) and finally motivate further research work in section 5.

2 Motivation and Background

The reason for the sparse usage of security requirements engineering methods and techniques in projects and the lack of maturity of security requirements specifications can be traced back to various issues. Frequently mentioned issues related to organizational and skill-related deficiencies are:

- Lack of security awareness [9]
- Feared negative business implications [10]
- Insufficient integration of security into development lifecycle processes [11, 6, 12]
- Lack of skills for security or security requirements engineering [10, 11].

Whilst these issues can only hardly be addressed by the research community, other problems are related to the current state of the art of security requirements engineering. Typically referred problems are:

Understandability and Usability of Security Requirements Engineering Methods and Frameworks. To identify and analyze the threat, attacker and misuse perspective various methods exist such as Microsofts STRIDE [13], Practical Threat Analysis (PTA) [14], Attack Path Analysis [15], Abuse Cases [16], Misuse Cases [17, 18], Anti Models [19] and many more. Moreover various modeling-oriented methods such as Tropos [20], CORAS [21], UMLSec [22] and Secure UML [23] were proposed to improve the

modeling of security requirements. However, all of these methods require certain security knowledge. They will only work sufficiently, if the security requirements analyst and the software architects know about the capabilities of attackers, available attacker tools on the market, typical threats, weaknesses and vulnerabilities. If one does not understand the mindset of an attacker, typical weaknesses and available exploits, the result of a threat analysis will likely be of only limited value. Although numerous publications state that security requirements engineering is important, only little concrete and specific advice is provided which can immediately be used in projects [24]. Security requirements engineering approaches are mainly tested on small scale, demanding for a broader external validation to demonstrate their effectiveness and usefulness in developing security solutions [12]. There is a demand amongst security practitioners that security requirements engineering methods must primarily be understandable and useable, besides being "formally pleasing or academically correct" [24].

Heterogeneous Definitions of Security Requirements. There is no common agreement what a security requirement is [12]. Moreover existing approaches do not agree on the extent to which the requirements should state concrete security measures [24]. This induces uncertainty among requirements engineering practitioners about the good practice of security requirements specifications and moreover leads to heterogeneous sets of security requirements ranging from a small set of high level security goals up to extensive lists of functional and non-functional security requirements or even concrete security measures.

Missing Consideration or Documentation of (trust) Assumptions. A trust assumption is "the firm belief in the competence of an entity to act dependably, securely and reliably within a specified context" [25]. "Trust assumptions can be used by a requirements engineer to help define and limit the scope of analysis and to document the decisions made during the process" [26]. "The level of trust has an approximate inverse relationship to the degree of risk" [25], which means that if everything and everyone is trusted no security threats and risks need to be incorporated, and therefore no security requirements need to be specified to address the threats and risks (assuming that no other external constraints or influences demand for security requirements). Unlike software-specific security requirements, assumptions cannot be enforced by the software-to-be [27], however they should be documented to ensure that organizations deploying the software in an environment know them and can develop environment-specific security requirements and measures to fulfill the trust assumptions. Unfortunately trust assumption or trust relations are seldom explicitly defined [28], since most of related work focusing on security requirements do not put much emphasis on trust [26].

Lack of Incorporation of the (intended) Operational Environment. "The system-to-be is in essence composite; it comprises both the software and its environment" [27]. Therefore security imposes not only constraints on software, but also on the environment [11]. If assumptions and constraints on the operational environment that have been made during the software development are not clearly specified and documented, they cannot be considered as input for secure deployment, operation and maintenance. Experience shows that assumptions and constraints resulting from the software design on the environmental (e.g. as part of a product sheet, a configuration or deployment guideline) are seldom explicitly specified.

In the following we investigate the interdependencies between the software security requirements, trust assumptions and the environment, by introducing a security dependency model for later discussion.

3 Interdependencies Between Software and Its Environment

3.1 Model to Illustrate the Interdependency Between Software and Its Environment

To sketch the interdependency between software and its environment, we use a security dependency model as displayed in figure 1. The model consists of three scope areas which are briefly introduced in the following. Since this paper is about software security requirements engineering, we focus on the interdependency between software and its environment (displayed as bold arrows) and do not explain the interdependency between the physical and technical, and the organizational environment (displayed as dotted arrows).

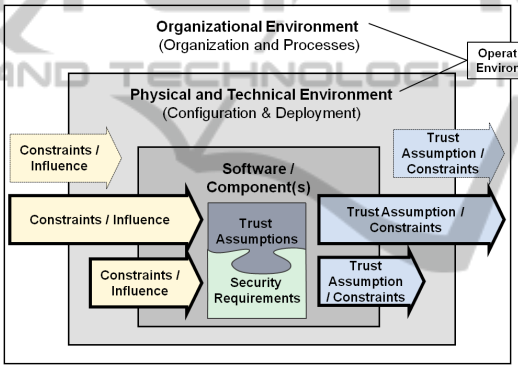

Fig. 1. Security interdependency model.

Software / Component(s). The heart of the model is the software itself. Typical software security aspects (sometimes also referred to as security quality factors [29]) are authentication, authorization/access control, session management, data at rest security or data in transit security. The desired outcome of the security requirements engineering phase is a set of security requirements and trust assumptions, which are typically part of a software requirements specification document. Software is deployed in an operational environment. We define an operational environment (similar to the definition of the term 'environment' in [30]) as procedures, conditions, and objects affecting the development, operation, and maintenance of a system. Since this definition is generic, we further subdivide the operational environment into a physical and technical environment and an organizational environment. The reason to split up the operational environment is motivated by the fact that constraints, influences and assumptions on and from software significantly vary between the two environmental scope areas. Furthermore the security measures being required to satisfy the security requirements can typically be assigned to one of both environment scope areas.

Physical and Technical Environment. The physical and technical environment consists of relevant physical and technical conditions and objects which are relevant for the software. Examples for objects within the physical and technical environment are servers, the areas, building and rooms where relevant system components are sited, client devices, network(s), other neighboring systems or services relevant for the software. Furthermore other security mechanisms and services protecting the infrastructure are in scope. Security aspects related to the system in its physical and technical environment are e.g. physical security, network security, secure software configuration, secure system configuration and hardening, malware protection.

Organizational Environment. The organizational environment includes relevant organizational aspects as well as processes and procedures required to securely set-up, operate, maintain and use the software. Examples for organizational aspects are e.g. the issuance of mandatory security policies and guidelines, the set-up of a security organization with clearly defined roles and responsibilities and the conduction of security training and awareness activities for employees and third party personnel. Examples for operational security related processes and procedures are user management, privilege management, key management, vulnerability and patch management, incident management and many more.[4]

Constraints / Influence on Software Security Requirements (bold inbound arrows). During security requirements engineering for software relevant constraints and influences from both, physical and technical environment, as well as the organizational environment need to be considered. The organizational environment constraints and influences the software security requirements by various means, namely:

- Stakeholder related needs such as functional needs, other non-security quality needs, business goals, asset protection needs, security goals
- Compliance obligations such as company-internal policies, legal, regulatory and normative obligations, accreditation, certification and auditing needs
- Domain-specific obligations such as sector or domain-specific characteristics or restrictions which specifically need to be considered)
- Threat / attacker / misuse perspective such as threats or attacks originating in the organizational environment, commonly referred weaknesses or vulnerabilities. The physical and technical environment constraints and influences the software security requirements through various parameters such as:
- Intended hosting model e.g. internal / on-premise, external / off-premise, combined
- Deployment model e.g. private, public, community, hybrid
- Client component-related aspects e.g. bring your own device vs. company owned and managed devices
- Implemented security services e.g. IDS, IPS, firewalls, network segmentation, malware protection (and many more)

Based on these characteristics the relevant threat, attacker and misuse perspective needs to be modeled and analyzed. It must be clear which threats and risk exist due to the phys-

[4] Please note that for simplicity reasons we intentionally omit non-operational security processes such as security risk management, security policy review and update, HR-related security aspects, etc.

ical and technical environment. Furthermore it must be analyzed how well established the assumed security mechanisms within the physical or technical environment are, and who or what is considered trusted.

Trust Assumptions / Constraints on the Environment (bold outbound arrows). Assumptions and constraints from software on the environment cannot be enforced by the software-to-be [27]. Nevertheless trust assumptions need to be fulfilled in the operational environment to guarantee that the taken software security measures are sufficient. Therefore trust assumptions being made during the software development as well as constraints on the environment must explicitly be specified, instead of simply assuming or hoping that they are satisfied e.g. by organizational norms and regulations, etc. [27]. Trust assumptions influencing the operational environment must be met by the customer to ensure that the software security requirements and measures are sufficient. Therefore specified trust assumptions must be handed over to and be analyzed by the customer within the concrete environment. Security requirements and associated security measures for the physical and operational requirement, as well as the organizational environment must be specified in order to meet these trust assumptions.

3.2 Example 1 - Influence from the Operational Environment on Software Security Requirements

To illustrate the influence from the operational environment on software security (requirements engineering) and to motivate the need for a comprehensive security analysis, we use a username / password-based user authentication mechanism as example.[5] Good practice and typical measures for password-based authentication are e.g. enforcement of a good password policy, enforcement of initial password change after first login, confidential transmission of passwords, etc. What might nevertheless lead to an unauthorized access even if the software-side user authentication mechanism is implemented well? A weakness or vulnerability in the physical or technical environment such as an unpatched server being exploited from the internet, theft of server components due to insufficient physical area or equipment protection. This may lead to a loss of usernames and associated passwords e.g. as part of username and passwords stored in a database and therefore compromise the security of the user authentication mechanism. Moreover, an attacker could exploit weaknesses in the organizational environment e.g. by using social engineering techniques to trick a service desk employee to reveal user password via telephone. This attack is possible even if we would assume a software-side authentication mechanism following good practices and a well secured physical and technical environment. An intuitive reaction would be to address the threats and mitigate the risk in the operational environment where they originate, but can these non-software related threats also be addressed by the software? In some cases the mentioned threats in the physical and technical, as well as the organizational environment can be either mitigated directly in the respective scope areas or by software-specific countermeasures. In our

[5] Please note that for simplicity reason we omit any potential stakeholder related needs, compliance obligations or domain-specific obligations, but concentrate on the threat, attacker and misuse perspective.

example, the risk of an unpatched system can be reduced by implementing countermeasures in the environment e.g. by hardening the server and protecting the physical area and equipment from unauthorized physical access. However the impact and thus also the risk of a hacked or stolen server component and the resulting loss of the password table can also be treated on software-level by implementing a mechanism to store passwords on the server only as salted hashes. Also the risk of social engineering attacks can be addressed on the one hand in the organizational environment, e.g. by performing security training and awareness for employees and on the other hand by the software security mechanisms, e.g. by providing a password reset and transmission functionality, in which passwords are never shown in clear text to a service desk employee and can only be transmitted via an encrypted email.[6] The example shows that attacks which are primarily threatening the operational environment and therefore not primarily the software can influence the software security requirements and software security mechanisms. If one could assume that the technical and organizational environment is fully trusted, passwords would probably not be stored as salted hashes. However this would imply that the operational environment is considered absolutely secure and thus hack proof and furthermore that administrators are allowed to see passwords in clear text, which is a very unrealistic assumption. The same applies for the organizational environment. If one could fully trust all service desk employees and they are assumed to withstand social engineering attacks, one would not have to design a sophisticated authentication mechanism but could verify the identity of a caller with adequate procedural measures. Furthermore the example indicates that the less predictable the variety of potential operational environments of a software-to-be are, the more difficult it becomes to specify valid trust assumptions and proper security requirements. The reason for that is that constraints and influences are not clear or may vary significantly. This on the one hand affects the number of potential threats, and on the other hand other influences and constraints need to be incorporated in the security requirements engineering phase to come to a proper security and trust assumption specification. The less concrete the influences from the operational environment on software are, the higher the variability and level of security should be.

3.3 Example 2 - Implications of Trust Assumptions / Constraints

We assume for our second example a client-server Web application using the Internet for the exchange of sensitive information. The software security aspect in our example that we want to focus on is ‘input validation’ in order to prevent various kinds of injection attacks. We assume the following trust assumptions (TA):

- TA1) Legitimate users do not enter malicious input data.
- TA2) Only legitimate users can access the system.

These two trust assumption are often implicitly assumed but nevertheless unrealistic. They limit the amount of threats and attacks which have to be considered during the software security requirements engineering phase, since a requirements engineer would

[6] Please note that of course there are multiple other countermeasures which can address the threat and mitigate the risk.

not have to specify any security requirements related to input validation. How do the trust assumptions influence the operational environment? The organization deploying the software somehow has to ensure that the trust assumptions are met in the physical and technical, as well as the organizational environment. One could say that trust assumptions and constraints must be treated as security requirements in the operational environment and must result in adequate organizational, physical and technical security measures. In our example, resulting security measures to be implemented by the customer in order to satisfy TA1 might be:[7]

- Conduction of security awareness and training for employees, as well as integration of a code of conduct as part of working contracts, and / or
- Implementation of a security proxy, ensuring user data input validation before input is forwarded to the server.

To satisfy the second trust assumption TA2, the customer might implement physical area and equipment protection measures for client devices and server, and prevent unauthorized logical access by using a physically isolated, access protected network. The example shows that trust assumptions are not only relevant for security during software development, but also play an important role as constraints on the physical, technical and organizational environment. Trust assumptions concerning the operational environment being made during software security requirements engineering are constraints that must be considered during security requirements engineering in the context of a software integration or deployment project. Therefore trust assumptions shall be made transparent to the software customer e.g. in a product sheet, configuration guideline, operating instruction or some kind of security documentation. Moreover not only the knowledge about implemented security-relevant functions in terms of 'what' and how they can be configured are of interest, but also the 'why' and relevant scenarios for which the security mechanism should be used. This facilitates on the one hand the evaluation of software security e.g. to check if a software product fits the desired purpose and is suitable for the intended environment. On the other hand it makes it easier to securely set-up, operate, maintain and use a software product by following the intended purpose and configurations. It is advisable that recommendations for the use of certain security functionality or configuration are associated with environmental constraints such as 'If client-server communication takes place over an untrusted network, the following software configuration is recommended' or 'Security of data in transit cannot be guaranteed by the software but must be ensured by additional network security mechanisms in the technical or physical environment.'

4 Summary and Conclusions

We started with the presentation of typical issues related to security requirements engineering, followed by the introduction and explanation of a security interdependency model. Influences from the operational environment on software security requirements,

[7] Please note that the first measure belongs the organizational environment and the second measures to the technical environment.

as well as implications of trust assumptions on the environment were presented and illustrated with two examples. On the basis of the security dependency model and the given examples we can draw following conclusions:

Dependency between security requirements and Trust Assumption. Implicit and explicit trust assumptions influence the scope of the security analysis, as well as the elicitation and specification of security requirements. Therefore security requirements and trust assumptions are interdependent and must be considered together to ensure an adequate level of risk.

Constraints / Influence on Software Security Requirements (bold inbound arrows). Both, the threats immediately affecting a software product, as well as threats in the operational environment of a software influence security requirements and associated software security mechanisms. The less predictable the potential operational environment of a software product is, the more difficult it becomes to specify valid trust assumptions and proper security requirements, since constraints and influences are unclear or may vary significantly. This lack of concrete constraints and influences from the operational environment on software demands for a higher variability and quality of security mechanisms in order fit as best as possible to different fields of application.

Influences of Security Requirements and Trust Assumptions on the Operational Environment (outbound bold arrows). Trust assumptions are not only relevant for security during software development, but also play an important role as constraints on the physical, technical and organizational environment. The approximate inverse relationship between the level of trust and the degree of risk might be tempting for a software development team to specify too many or too extensive trust assumptions, since they save a lot of effort with respect to security analysis and the specification of software security requirements and measures. However, risks not being analyzed and mitigated on software-level are passed on to the operational environment. Some of these risks or shortcomings in software security can potentially be rectified by other or additional security mechanisms in the operational environment. Nevertheless there are other cases where the rectification of security weaknesses or vulnerabilities of software cannot or only at a very high expense be rectified.

5 Future Research Perspectives

5.1 Development of a Method for Reuse-oriented Engineering of Security Requirements and Trust Assumptions

We believe that for many security aspects typical threats, generic requirements, and guidelines can be elaborated and brought together into a consistent and traceable relation on a generic level. The resulting information and knowledge base is intended to be customizable for the needs of an organization, especially with regard to the raw requirements, influences and constraints from their customer and environment. The resulting structure can be reused within the organization for the elicitation and refinement of security requirements and trust assumptions. The scope of the aspects is not only limited to software or product-related security aspects, but should also incorporate security

technical, physical and organizational aspects of systems in their operational environment. The research questions that we pursue regarding reuse-oriented engineering of security requirements and trust assumptions are:

- What are suitable scope areas and the most prevalent security aspects that can be reasonably covered in a reuse-oriented SRE method and how do they influence each other?
- Is it possible to create a limited (but nevertheless useful) list of common threats and weaknesses?
- How well can threats and weaknesses be associated with recommended security requirements and/or security measures?
- How can constraints and influences between the scope areas and security aspects be made transparent?
- How detailed can or should security requirements be?
- How can the specification of trust assumptions be incorporated into the reuse-oriented SRE approach?
- Which information and knowledge about the solution space can / should be provided to support the support the interaction with the security architect and the refinement of requirements from high-level to a more detailed level?

We are fully aware that a reuse-oriented approach of security requirements engineering can only reach a baseline security level and that it has to be on a generic level to be universally applicable. However, security requirements specification could be supported, and effort can be reduced through the reuse and refinement of existing information and knowledge sources.

5.2 Provisioning of a Framework or Process for the Integration of a Reuse-oriented Method

In addition to the method for reuse-oriented security requirements, a corresponding framework or process needs to be developed, explaining how the reuse-oriented method needs to be customized for, and used within an organization. Besides the existing, undoubted security requirements activities of existing frameworks and processes, the following steps should be included.

- Development of an organization-specific, reuse-oriented SRE framework incorporating relevant SRE sources such as common threats and weaknesses, generic security requirements, as well as guidelines etc.
- Incorporation of the specification of trust assumptions in addition to security requirements
- Recommendation or provisioning of risk-based security requirements engineering methods to accompany or complement the reuse-oriented method

To the best of our knowledge there is no requirements engineering framework incorporating all of the before mentioned aspects.

Acknowledgement

We would like to thank all reviewers of this paper who provided valuable feedback, especially Stefan Seltzsam and Steffen Fries for their support.

References

1. Khaled El Emam and A. Günes Koru. A replicated survey of it software project failures. IEEE Software, 25(5):84–90, 2008.
2. Department of Homeland Security. Requirements analysis for secure software, 2012.
3. Andy Greenberg. A tax on buggy software, 2008. http://www.forbes.com/2008/06/26/rice-cyber-security-tech-security-cx_ag_0626rice.html. Visited on January 15th, 2014.
4. Barry W. Boehm. Software engineering economics. Prentice-Hall advances in computing science and technology series. Prentice-Hall, Englewood Cliffs and N.J, 1981.
5. G. McGraw. Testing for security during development: why we should scrap penetrate-and-patch. Aerospace and Electronic Systems Magazine, IEEE, 13(4):13–15, 1998.
6. Golnaz, Elahi, Yu, Eric, Tong Li, Lin Liu. Security requirements engineering in the wild: A survey of common practices. In Proceedings of the 35th Annual IEEE International Computer Software and Applications Conference, Proceedings - International Computer Software & Applications Conference, pages 314–319, Danvers, 2011. IEEE.
7. John Wilander and Jens Gustavsson. Security requirements – a field study of current practice, 2005.
8. Nancy R. Mead. Security requirements engineering, 2006. https://buildsecurityin.us-cert.gov/bsi/articles/best-practices/requirements/243-BSI.html. Visited on January 15th, 2014.
9. Theodore Winograd, Holly Lynne McKinley, Lyndon Oh, Michael Colon, Thomas McGibbon, Elaine Fedchak, and Robert Vienneau. Software security assurance: A State-of-the Art Report (SOAR). Information Assurance Technology Analysis Center, Herndon and Virginia, 2007.
10. Donald G. Firesmith. Engineering security requirements. Journal of Object Technology, vol. 2, no. 1,, pages 53–68, 2003.
11. Haralambos Mouratidis, Paolo Giorgini, and Gordon Manson. When security meets software engineering: a case of modelling secure information systems. Information Systems, 30(8):609–629, 2005.
12. Eric Dubois and Haralambos Mouratidis. Guest editorial: security requirements engineering: past, present and future. Requirements Engineering, 15(1):1–5, 2010.
13. Frank Swiderski and Window Snyder. Threat modeling. Microsoft Press, Redmond and Wash, 2004.
14. PTA Technologies. Practical threat analysis for information security experts. http://www.ptatechnologies.com/default.htm. Visited on January 15th, 2014.
15. Yue Chen. Software security economics and threat modeling based on attack path analysis; a stakeholder value driven approach. University of Southern California. Libraries, 2007.
16. J. McDermott and C. Fox. Using abuse case models for security requirements analysis. In Computer Security Applications Conference, 1999. (ACSAC '99) Proceedings. 15th Annual, pages 55–64, 1999.
17. Ian F. Alexander. Initial industrial experience of misuse cases in trade-off analysis. In Proceedings of the 10th Anniversary IEEE Joint International Conference on Requirements Engineering, RE '02, pages 61–70, Washington and DC and USA, 2002. IEEE Computer Society.

18. Guttorm Sindre and Andreas L. Opdahl. Eliciting security requirements with misuse cases. Requir. Eng, 10(1):34–44, 2005.
19. Axel van Lamsweerde. Elaborating security requirements by construction of intentional anti-models. In Proceedings of the 26th International Conference on Software Engineering, ICSE '04, pages 148–157, Washington and DC and USA, 2004. IEEE Computer Society.
20. Jaelson Castro, Manuel Kolp, and John Mylopoulos. Towards requirements-driven information systems engineering: The tropos project. Inf. Syst., 27(6):365–389, 2002.
21. F. Braber, I. Hogganvik, M. S. Lund, K. Stølen, and F. Vraalsen. Model-based security analysis in seven steps — a guided tour to the coras method. BT Technology Journal, 25(1):101–117, 2007.
22. Jan Jürjens. Towards development of secure systems using umlsec. Fundamental Approaches to Software Engineering, volume 2029 of Lecture Notes in Computer Science, pages 187–200. Springer Berlin Heidelberg, 2001.
23. Torsten Lodderstedt, David Basin, and Jürgen Doser. Secureuml: A uml-based modeling language for model-driven security. ≪UML≫ 2002 — The Unified Modeling Language, volume 2460 of Lecture Notes in Computer Science, pages 426–441. Springer Berlin Heidelberg, 2002.
24. I. A Tondel, M. G Jaatun, and P. H Meland. Security requirements for the rest of us: A survey. Software, IEEE (Volume: 25 , Issue: 1), pages 20–27, 2008.
25. T. Grandison and M. Sloman. A survey of trust in internet applications. Communications Surveys & Tutorials, IEEE, 3(4):2–16, 2000.
26. Charles B. Haley, Robin C. Laney, Jonathan D. Moffett, and Bashar Nuseibeh. Picking battles: The impact of trust assumptions on the elaboration of security requirements. Trust Management, volume 2995 of Lecture Notes in Computer Science, pages 347–354. Springer Berlin Heidelberg, 2004.
27. A. van Lamsweerde. Goal-oriented requirements engineering: a guided tour. In Fifth IEEE International Symposium on Requirements Engineering, pages 249–262, 27-31 Aug. 2001.
28. John Viega and Gary McGraw. Building secure software: How to avoid security problems the right way. Addison-Wesley professional computing series. Addison-Wesley, Boston, 2002.
29. Donald G. Firesmith. Analyzing and specifying reusable security requirements. Journal of Object Technology, (Vol. 3, No. 1):61–75, 2004.
30. NIST. Glossary of key information security terms. U.S. Deptartment of Commerce, National Institute of Standards and Technology, [Gaithersburg and Md.], 1 edition, 2011.