SREP: A Proposal for Establishing Security Requirements for the Development of Secure Information Systems

Daniel Mellado[1], Eduardo Fernández-Medina[2] and Mario Piattini[2]

[1] Ministry of Labour and Social Affairs; Management Organism of Information Technologies of the Social Security; Quality, Auditing and Security Institute; Madrid, Spain

[2] ALARCOS Research Group, Information Systems and Technologies Department, UCLM-Soluziona Research and Development Institute, University of Castilla-La Mancha Paseo de la Universidad 4, 13071 Ciudad Real, Spain.

Abstract. Nowadays, security solutions are mainly focused on providing security defences, instead of solving one of the main reasons for security problems that refers to an appropriate Information Systems (IS) design. In this paper a proposal for establishing security requirements for the development of secure IS is presented. Our approach is an asset-based and risk-driven method, which is based on the reuse of security requirements, by providing a security resources repository, together with the integration of the Common Criteria into traditional software lifecycle model, so that it conforms to ISO/IEC 15408. Starting from the concept of iterative software construction, we will propose a micro-process for the security requirements analysis, that is repeatedly performed at each level of abstraction throughout the incremental development. In brief, we will present an approach which deals with the security requirements at the first stages of software development in a systematic and intuitive way, and which conforms to ISO/IEC 17799:2005.

1 Introduction

Present-day information systems are vulnerable to a host of threats. Moreover, with the increase of the complexity of applications and services, there is a correspondingly greater chance of suffering from breaches in security [20]. In our contemporary Information Society, depending as it does on a huge number of software systems which have a critical role, it is absolutely vital that IS are properly ensured from the very beginning [1, 12], due to the potential losses faced by organizations that put their trust in all these IS.

As we know, the principle which establishes that the building of security at the early stages of the development process is cost-effective and also brings about more robust designs is widely-accepted [10]. The biggest problem, however, is that in the majority of software projects security is dealt with when the system has already been designed and put into operation. On many occasions, this is due to an inappropriate

Mellado D., Fernández-Medina E. and Piattini M. (2006).
SREP: A Proposal for Establishing Security Requirements for the Development of Secure Information Systems.
In *Proceedings of the 4th International Workshop on Security in Information Systems*, pages 135-145

management of the specification of the security requirements of the new system, since the stage known as the requirements specification phase is often carried out with the aid of just a few descriptions, or the specification of objectives that are put down on a few sheets of paper [4]. Added to this, the actual security requirements themselves are often not well understood. This being so, even when there is an attempt to define security requirements, many developers tend to describe design solutions in terms of protection mechanisms, instead of making declarative propositions regarding the level of protection required [5].

A very important part in the software development process for the achievement of secure software systems is that known as security Requirements Engineering (RE). Which provides techniques, methods and norms for tackling this task in the IS development cycle. It should involve the use of repeatable and systematic procedures in an effort to ensure that the set of requirements obtained is complete, consistent and easy to understand and analyzable by the different actors involved in the development of the system [11]. A good requirements specification document should include both functional requirements (related to the services which the software or system should provide), and non-functional (related to what are known as features of quality, performance, portability, security, etc) [4]. As far as security is concerned, it should be a consideration throughout the whole development process, and it ought to be defined in conjunction with the requirements specification [14].

In this paper, we will present the Security Requirements Engineering Process (SREP), which explains how to integrate the security requirements into the software engineering process in a systematic and intuitive way. Our approach is based on the integration of the Common Criteria (CC) into the traditional software lifecycle model, together with the reuse of security requirements which are compatible with the CC Framework subset. In addition, in order to support this method and make this task easy, we will propose the use of several concepts and techniques: a security resources repository (with assets, threats, requirements, etc), the use of UMLSec [17], misuse cases [18], threat/attack trees, and security uses cases [6]. Overall, we will propose an asset-based and risk-driven method for the establishment of security requirements. The remainder of the paper is set out as follows: in section 2, we will outline an overview of our Security Requirements Engineering Process. Section 3 will present the main characteristics of this methodology. Next, in section 4, we will explain the security resources repository. And, we will describe the process model in section 5. Lastly, our conclusions and further research are set out in section 6.

2 A General Overview of SREP

The Security Requirements Engineering Process (SREP) is an asset-based and risk-driven method for the establishment of security requirements in the development of secure Information Systems. Basically, this process describes how to integrate the CC into the traditional software lifecycle model together with the use of a security resources repository to support reuse of security requirements (modelled with UMLSec, or expressed as security use cases or as plain text with formal specification), assets, threats (which can be expressed as misuse cases, threat/attack trees, UMLSec dia-

grams) and countermeasures. The focus of this methodology seeks to build security concepts at the early stages of the development lifecycle.
We will show a brief outline of SREP below in Fig. 1.

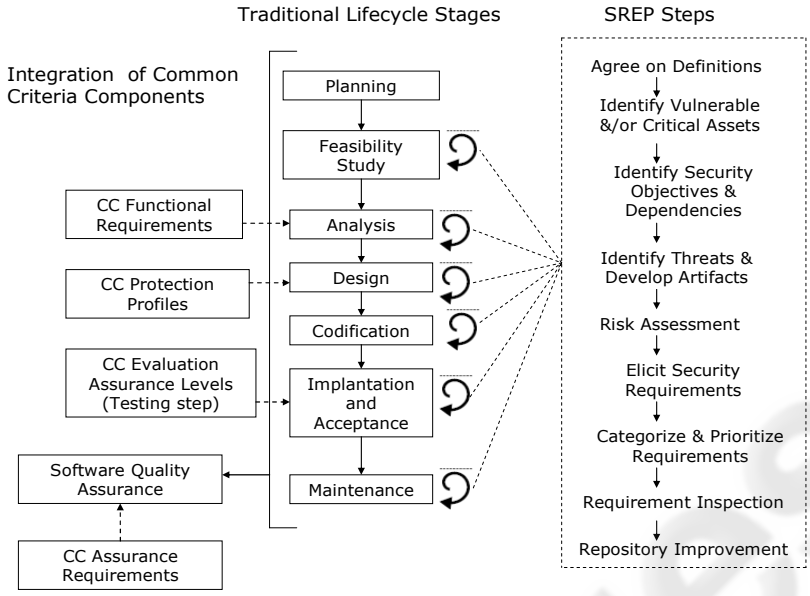

Fig. 1. SREP overview.

As it is described in Fig. 1, the core of SREP is a micro-process, made up of nine steps which are repeatedly performed at each stage of the lifecycle. At the same time, the CC Components are introduced into the software lifecycle, so that SREP uses different CC Components according to the stage, although the Software Quality Assurance (SQA) activities are performed along all the stages of the software development lifecycle. And it is in these SQA activities where the CC Assurance Requirements might be incorporated into. A more detailed explanation of the integration of CC Components and SREP into the IS development lifecycle is presented in the following sections.

3 Characteristics of SREP

In general terms the main characteristics of SREP are:

- Iterative and incremental. The model chosen for SREP is iterative and incremental, thus the security requirements evolve along the lifecycle; for instance, during the design, the specification could be enriched with requirements related to the technological environment and its associated countermeasures. The core concept is the use of a micro-process for the security requirements analysis [2], made up of nine steps, which are repeatedly performed at each level of abstrac-

tion throughout the incremental development. Each iteration accomplishes all the steps defined within SREP, and each output from a complete iteration improves and refines the Security Requirements Specification by adding, correcting or specifying/detailing security requirements.

- It facilitates the reusability. We proposed a security resources repository and a meta-model for it (based on Sindre, Firesmith and Opdahl approach [18]). The purpose of development with requirements reuse is to identify descriptions of systems that could be used (either totally or partially) with a minimal number of modifications, thus reducing the total effort of development [3]. Moreover, reusing security requirements helps us increase their quality: inconsistency, errors, ambiguity and other problems can be detected and corrected for an improved use in subsequent projects [19]. Thereby, it will guarantee us the fastest possible development cycles based on proven solutions.
- It facilitates the traceability of the security requirements along the levels of abstraction, thanks to the structure of the repository.
- It supports and includes concepts and techniques within the scope of Security Requirement Engineering and Risk Management and Analysis, such as UMLSec, security use cases, misuse cases, threat/attack trees.
- Finally, it conforms to several standards within the scope of Requirement Engineering and Security Management, like ISO/IEC 17799:2005 and ISO/IEC 15408.

3.1 SREP Compliance with Standards

SREP conforms to ISO/IEC 17799:2005 recommendation with regard to security requirements. It says that "Security requirements should be identified and agreed prior to the development and/or implementation of information systems. All security requirements should be identified at the requirements phase of a project and justified, agreed, and documented as part of the overall business case for an information system". And this is exactly what SREP proposes to do.

Moreover, we take into account the IEEE 830-1998 standard, so that the step of "Requirements Inspection" of our micro-process for the security requirements analysis verifies whether the security requirements conform to this standard. Because according to the IEEE 830-1998 standard, a requirement of quality has to be correct, unambiguous, complete, consistent, ranked for importance and/or stability, verifiable, modifiable, and traceable. Therefore all these factors are verified at the end of each iteration of the micro-process, just before the "Repository Improvement" step.

The CC (ISO/IEC 15408) is the standard requirements catalogue for the evaluation of security critical systems. Using the CC, a large number of security requirements within the system itself and in the system development can be defined. And the CC scheme can be introduced into the software lifecycle of new and existing applications to meet stricter security requirements. So, we propose to introduce it. This can be accomplished, according to Kam [9], by: integrating CC functional requirements into the Software Requirements Specification; integrating CC assurance requirements into Software Quality Assurance (SQA) activities; introducing EALs (Evaluation Assurance Levels) into the software test plan; and introducing CC Protection Profiles into

architectural design. Although a detailed explanation of the latter ones is outside the scope of this paper.

4 The Security Resources Repository

SREP is based on several current techniques, which deal with security requirements, in order to make it easy the task of dealing with security requirements in the first stages of software development in a systematic and intuitive way. The main ones are exposed below.

- *UMLSec* allows us to express security-related information within the diagrams in a UML system specification, thereby it aims to be more integrated with the artefacts produced during the development process. The extension is given in the form of a UML profile using the standard UML extension mechanisms. Stereotypes are used together with tags to formulate security requirements and assumptions on the system environment; constraints give criteria that determine whether the requirements are met by the system design [17]. We used UMLSec to specify the security requirements, and it is a complement method to security use cases.
- *Security Use Cases* are a technique that we used in order to specify the security requirements that the application must fulfil to be able to successfully protect itself from its relevant security threats [6]. And they are driven by misuse cases.
- *Misuse Cases* are a specialized kind of use cases that are used to analyze and specify security threats [6]. They are the inverse of a use case, a function that the system should not allow. In more detail it might be defined as a completed sequence of actions which results in losses for the organization or some specific stakeholder [18]. In our approach they drive the security use cases, and threats are expressed as misuse cases.

The *Security Resources Repository* (SRR) stores all the reusable elements which can be used by the analysts. The repository understands the concepts of domains and profiles [19]. The former consists of belonging to a specific application field or functional application areas, such as e-commerce. The latter consists of a homogeneous set of requirements which can be applied to different domains, as for example personal data privacy legislation. We propose to set an attribute for each element of the SRR that shows in which/s domain/s it can be used. On the other hand, the profiles, together with the CC Protection Profiles, are stored as standardized subsets of specific security requirements together with its related elements of the SRR (threats, etc.). In brief, each domain or profile is a view of the global SRR.

Furthermore, the elements included in the SRR have been generically established by using parameter-based mechanisms, such as reusable parameterized templates. Although there are also non-parameterized templates and checklists, such as asset checklists. In addition, each security requirement and its specification are labelled as System Requirement (IEEE Std. 1233 and IEEE Std. 1207.1) or Software Requirement (IEEE Std. 830-1998) [19].

A meta-model, which is an extension of the meta-model for repository proposed by Sindre, G., D.G. Firesmith, and A.L. Opdahl [18], showing the organization of the SRR is exposed below in Fig. 2.

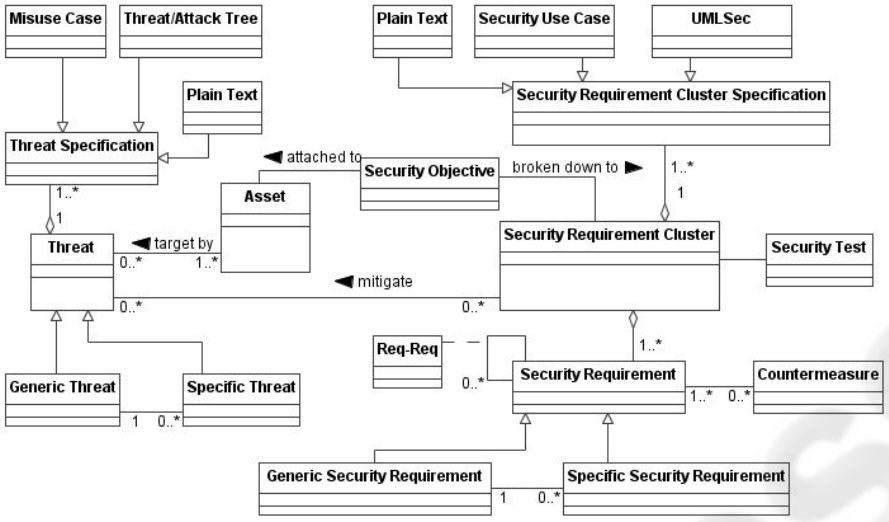

Fig. 2. Meta-model for security resources repository.

As it is presented above, it is an asset-driven as well as a threat-driven meta-model, because the requirements can be retrieved via assets or threats. Next, we will outline the most important and/or complex aspects of the meta-model.

- 'Generic Threat' and 'Generic Security Requirement' describe independently of particular domains. And they can be represented as different specifications, thanks to the elements 'Threat Specification' and 'Security Requirement Cluster Specification'.
- 'Security Requirement Cluster' is a set of requirements that work together in satisfying the same security objective and mitigating the same threat. We agree with Sindre, G., D.G. Firesmith, and A.L. Opdahl [18] that, in many cases, it is a bigger and more effective unit of reuse.
- The 'Req-Req' relationship allows an inclusive or exclusive trace between requirements. An exclusive trace between requirements means that they are mutually alternative, as for example that they are in conflict or overlapping. Whereas, an inclusive trace between requirements means that to satisfy one, another/other/s is/are needed to be satisfied.

In addition, there could have been links further on to design level specifications, security test cases, countermeasures, etc. Due to the fact that our proposed model process is based on the concept of iterative software construction, as we will explain in the next section.

Finally, we would like to point out the fact that using the CC, a large number of security requirements on the system itself and on the system development can be defined. Nevertheless, the CC does not give methodological support, nor contain security evaluation criteria pertaining to administrative security measures not directly

related to the IS security measures. However, it is known that an important part of the security of an IS can be often achieved through administrative measures. Therefore, according to ISO/IEC 17799:2005, we propose to include legal, statutory, regulatory, and contractual requirements that the organization, its trading partners, contractors, and service providers have to satisfy, and their socio-cultural environment. After converting these requirements into software and system requirements format, these would be the initial subset of security requirements of the SRR. Moreover, if the organization has any activity in Spain we propose that the SRR contains all the requirements taken from MAGERIT, the Spanish public administration risk analysis and management method, which conforms to ISO 15408, as well as lists of assets, threats and countermeasures. This way, it will constitute a profile which conforms to Spanish security and data privacy protection legislation.

5 Process Model

According to Kotonya and Sommerville [11], RE basically comprises the phases/steps of i) requirements elicitation, ii) requirements analysis and negotiation, iii) requirements documentation and iv) requirements validation. Starting from the concept of iterative software construction, we propose a micro-process for the security requirements analysis, made up of nine steps, which are repeatedly performed at each level of abstraction throughout the incremental development. As the Security Requirements Specification document will evolve during the rest of the life cycle; for instance, during the design, the specification could be enriched with requirements related to the technological environment. Moreover, each security requirement can be traced along the levels of abstraction, and also, as the model understands the concepts of profiles and domains (that may be made up of elements of different abstraction level), they will be analysed by stakeholders who have the best knowledge or/and the responsibility of the domain. Furthermore, we agree with Nuseibeh [16] that the RE and architecture design processes are concurrent and influence each other.

The nine steps (based on [18] and [13]) of the micro-process for the security requirements analysis are presented below:

- Step 1: Agree on definitions
- Step 2: Identify vulnerable and/or critical assets
- Step 3: Identify security objectives and dependencies
- Step 4: Identify threats and develop artefacts.
- Step 5: Risk assessment
- Step 6: Elicit security requirements
- Step 7: Categorize and prioritize requirements
- Step 8: Requirements inspections
- Step 9: Repository improvement

(Steps 2–5: i) Requirements elicitation; Step 6: ii) Requirements analysis and negotiation; Step 7: ii) Requirements analysis and negotiation; Step 8: iii) Documentation and iv)Validation; Step 9: iii) Documentation and iv)Validation)

- *Step 1: Agree on definitions*. The first task for the organization is to agree upon a common set of security definitions, along with the definition of the organizational security objectives. The candidate definitions will be from IEEE and other stan-

dards. And it is important the participation of the stakeholder and the requirements team in this step, which will be performed only once.

- *Step 2: Identify vulnerable and/or critical assets*. This is where the SRR is used for the first time. It consists of the identification of the different kinds of valuable or critical assets as well as vulnerable assets by the requirements engineer, who can be helped by using:
  - Lists of assets of the SRR, where the analyst/requirements engineer can search by domains, even he/she can select a concrete profile.
  - Functional requirements.
  - Interviews with the stakeholders.
- *Step 3: Identify security objectives and dependencies*. At this step the requirements engineer can use the SRR too, because each asset has security objectives attached. For each asset identified at the previous step, the requirements engineer selects the appropriate security objectives for the asset and identifies the dependencies between them. Security goals are expressed by specifying the necessary security level as a probability, and they are also specified in terms of likely attacker types.
- *Step 4: Identify threats and develop artefacts*. Each asset is targeted by threat/s that can prevent the security objective from being achieved. First of all, it is necessary to find all the threats that target these assets with the help of the SRR. In addition, it could be necessary to develop artefacts (such as misuse cases or attack trees diagrams or UMLSec use cases and classes or sequence/state diagrams) to develop new specific or generic threat or requirement. As it is necessary to look for threats that are not linked/related to the assets of the repository, because it is possible that some security goals and assets may have been forgotten, or these threats may not be introduced in the repository yet.
- *Step 5: Risk assessment*. Risk must be normally determined from application to application. The final goal to achieve is the 100% risk acceptance. Firstly, it is necessary to assess whether the threats are relevant according to the security level specified by the security objectives. Then we have to estimate the security risks based on the relevant threats, their likelihood and their potential negative impacts. Whatever methodology can be used such as for example MAGERIT in Spain. Thereby, this assessment allows us to discover how the organization's risk tolerance is affected with regard to each threat. At this step, the requirements engineer should be helped by a risk expert and stakeholders.
- *Step 6: Elicit security requirements*. Here, the SRR is used again. For each threat retrieved from the repository, one or more associated clusters of security requirements may be found. The requirements engineer must select the suitable security requirements or the suitable cluster of security requirements that mitigate the threats at the necessary levels with regard to the risk assessment. However, additional requirements or clusters of requirements may be found by other means. Moreover, it might be specified the security test for each security requirement cluster, as well as an outline of the countermeasures for each security requirement, although they are refined at the design stage. Nevertheless, we agree with Firesmith [5] in the fact that care should be taken to avoid unnecessarily and prematurely architectural mechanisms specification.

- *Step 7: Categorize and prioritize requirements*. Each requirement is categorized and prioritized in a qualitative ranking in a way that the most important requirements (in terms of impact and likelihood) are handled first. This task is performed by the requirements engineer and other kind of specialists (if it is needed). After all, the requirements documentation is written.
- *Step 8: Requirements inspection*. Requirements inspection is carried out in order to validate the requirements, the modified model elements and the new generated model elements. Its aim is to review the quality of the team's work and deliverables. And it is performed by the inspection team. So, it is used as a sanity check.
- *Step 9: Repository improvement*. First of all, the validation of the requirements with the participation of the stakeholders and requirements engineer is carried out. Then, those new model elements (threats, requirements, etc…) found throughout the development of the previous steps and which are considered as likely to be used in forthcoming applications are introduced into the SRR. Furthermore, the model elements already in the repository could be modified in order to improve their quality.

Finally, at the same time, as we integrate into these nine steps the CC security functional requirements, we propose to outline the EALs in the software test plan and then verify them during test execution. And parallely, we proposed to introduce the CC security assurance requirements into SQA activities, like quality control, defect prevention and defect removal activities [9]. Consequently, the configuration management plan is the first activity that is explicitly required to fulfil the CC security assurance requirements, in order to achieve that the CC are integrated into the traditional development lifecycle.

6 Conclusions and Further Research

In our present so-called Information Society the increasingly crucial nature of IS with corresponding levels of new legal and governmental requirements is obvious. For this reason, the development of more and more sophisticated approaches to ensuring the security of information is becoming a necessity. Information Security is usually only tackled from a technical viewpoint at the implementation stage, even though it is an important aspect. We believe it is fundamental to deal with security at all stages of IS development, especially in the establishment of security requirements, since these form the basis for the achievement of a robust IS.

Consequently, we present an approach that deals with the security requirements at the first stages of software development, which is based on the reuse of security requirements, by providing a Security Resources Repository (SRR), together with the integration of the Common Criteria into traditional software lifecycle model. Moreover, it conforms to ISO/IEC 15408 and ISO/IEC 17799:2005. And we work towards 100% risk acceptance, because despite the best efforts of security researchers, it is impossible to guarantee 100% security [15]. Starting from the concept of iterative software construction, we propose a micro-process for the security requirements analysis, made up of nine steps, which are repeatedly performed at each level of abstraction throughout the incremental development. Finally, one of the most relevant

aspects is the fact that this proposal integrates other approaches, such as SIREN [19], UMLSec [17], security use cases [6] or misuse cases [18]. And it is also compatible with WSSecReq (Web Services Security Requirements) stage of the PWSSec (Web Services Security Development Process) process [7], as well as SREP might incorporate into its SRR the catalogue of security requirements template for web services based on SIREN, which Gutierrez et al. propose in [8].

Further work is also needed to provide a CARE (Computer-Aided Requirements Engineering) tool which supports the process, as well as a refinement of the theoretical approach by proving it with a real case study.

Acknowledgements

This paper has been produced in the context of the DIMENSIONS (PBC-05-012-2) Project of the Consejería de Ciencia y Tecnología de la Junta de Comunidades de Castilla- La Mancha along with FEDER and the CALIPO (TIC2003-07804-CO5-03) and RETISTIC (TIC2002-12487-E) projects of the Dirección General de Investigación del Ministerio de Ciencia y Tecnología.

References

1. Baskeville, R., The development duality of information systems security. Journal of Management Systems, 1992. 4(1): p. 1-12.
2. Breu, R. and Innerhofer–Oberperfler, F., Model based business driven IT security analysis. 2005: SREIS 2005.
3. Cybulsky, J. and Reed, K., Requirements Classification and Reuse: Crossing Domains Boundaries. ICSR'2000, 2000: p. 190-210.
4. Fernández-Medina, E., Moya, R., and Piattini Velthus, M., Gestión de Requisitos de Seguridad, in Seguridad de las Tecnologías de la Información "La construcción de la confianza para una sociedad conectada", AENOR, Editor. 2003. p. pp 593-618.
5. Firesmith, D.G., Engineering Security Requirements. Journal of Object Technology, 2003. 2(1): p. 53-68.
6. Firesmith, D.G., Security Use Cases. 2003: Journal of Object Technology. p. 53-64.
7. Gutierrez, C., Fernández-Medina, E., and Piattini, M., PWSSec: Process for Web Services Security. IEEE ICWS'05, 2005.
8. Gutiérrez, C., Moros, B., Toval, A., Fernández-Medina, E., and Piattini, M., Security Requirements for Web Services based on SIREN. Symposium on Requirements Engineering for Information Security (SREIS-2005), together with the 13th IEEE International Requirements Engineering Conference – RE'05, 2005.
9. Kam, S.H., Integrating the Common Criteria Into the Software Engineering Lifecycle. IDEAS'05, 2005: p. 267-273.
10. Kim., H.-K., Automatic Translation Form Requirements Model into Use Cases Modeling on UML. ICCSA 2005, 2005: p. 769-777.
11. Kotonya, G. and Sommerville, I., Requirements Engineering Process and Techniques. Hardcover ed. 1998. 294.
12. McDermott, J. and Fox, C. Using Abuse Case Models for Security Requirements Analysis. in Annual Computer Security Applications Conference. 1999. Phoenix, Arizona.

13. Mead, N.R. and Stehney, T. Security Quality Requirements Engineering (SQUARE) Methodology. in Software Engineering for Secure Systems (SESS05), ICSE 2005 International Workshop on Requirements for High Assurance Systems. 2005. St. Louis.
14. Mouratidis, H., Giorgini, P., Manson, G., and Philp, I. A Natural Extension of Tropos Methodology for Modelling Security. in Workshop on Agent-oriented methodologies, at OOPSLA 2002. 2003. Seattle, WA, USA.
15. Myagmar, S., J. Lee, A., and Yurcik, W., Threat Modeling as a Basis for Security Requirements. 2005: SREIS 2005.
16. Nuseibeh, Weaving Together Requirements and Architectures. IEEE Computer, 2001: p. 115-117.
17. Popp, G., Jürjens, J., Wimmel, G., and Breu, R., Security-Critical System Development with Extended Use Cases. 2003: 10th Asia-Pacific Software Engineering Conference. p. 478-487.
18. Sindre, G., Firesmith, D.G., and Opdahl, A.L. A Reuse-Based Approach to Determining Security Requirements. in Proc. 9th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'03). 2003. Austria.
19. Toval, A., Nicolás, J., Moros, B., and García, F., Requirements Reuse for Improving Information Systems Security: A Practitioner's Approach. 2001: Requirements Engineering Journal. p. 205-219.
20. Walton, J.P., Developing a Enterprise Information Security Policy. 2002, ACM Press: Proceedings of the 30th annual ACM SIGUCCS conference on User services.