Applying a Security Requirements Engineering Process

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

[1] Ministry of Labour and Social Affairs; Information Technology Center of the National Social Security Institute; Madrid, Spain
Daniel.Mellado@alu.uclm.es
[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
{Eduardo.FdezMedina, Mario.Piattini}@uclm.es

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 fact, requirements engineering often neglects enough attention to security concerns. In this paper it will be presented a case study of our proposal, called SREP (Security Requirements Engineering Process), which is a standard-centred process and a reuse-based approach which deals with the security requirements at the earlier stages of software development in a systematic and intuitive way by providing a security resources repository and by integrating the Common Criteria into the software development lifecycle. In brief, a case study is shown in this paper demonstrating how the security requirements for a security critical IS can be obtained in a guided and systematic way by applying SREP.

1 Introduction

Present-day information systems are vulnerable to a host of threats. What is more, with increasing 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 ensured as being safe right from the very beginning [1, 13].

As we know, the principle which establishes that the building of security into the early stages of the development process is cost-effective and also brings about more robust designs is widely-accepted [9]. 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. 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 [4].

D. Gollmann, J. Meier, and A. Sabelfeld (Eds.): ESORICS 2006, LNCS 4189, pp. 192–206, 2006.

A very important part of the achieving of secure software systems in the software development process is that known as Security Requirements Engineering, 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 [10]. A good requirement specification document should include both functional requirements and non-functional. 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 [16].

After having performed a comparative analysis of several relevant proposals of IS security requirements, as those of Yu 1997 [21], Toval et al. 2001 [19], Popp et al. 2003 [17], Firesmith 2003 [5], Breu, et al. 2004 [3], etc. in [15], we concluded that those proposals did not reach the desired level of integration into the development of IS, nor are specific enough for a systematic and intuitive treatment of IS security requirements at the early stages of software development. In addition, as yet, only few works (such as the article of Massacci et al. [12]) describes complex case studies which really cope with the complexity required by security standards, such as ISO/IEC 17799 [7] and ISO/IEC 15408 [8], compliance. Therefore, in this paper we briefly present the Security Requirements Engineering Process (SREP) [14] along with a case study of this proposal, which describes how to integrate security requirements into the software engineering process in a systematic and intuitive way. In order to achieve this goal, our approach is based on the integration of the Common Criteria (CC) [8] into the software lifecycle model, because the CC helps us deal with the security requirements along all the IS development lifecycle, together with the reuse of security requirements which are compatible with the CC Framework subset. In addition this proposal integrates other approaches such as UMLSec [17], security use cases [5] or misuse cases [18]. The remainder of this paper is set out as follows: in section 2, we will describe SREP. We will present the case study of SREP in section 3. Next, in section 4 it is presented the lessons learned. Lastly, our conclusions will be set out in section 5.

2 SREP: Security Requirements Engineering Process

To describe our proposal, we will rely on the process description patterns used in the Unified Process (UP) [2], since it is a use-case and risk driven, architecture-centric, iterative and incremental development process frame-work that leverages the Object Management Group's (OMG) UML and that is compliant with the OMG's Software Process Engineering Metamodel (SPEM).

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 software lifecycle model together with the

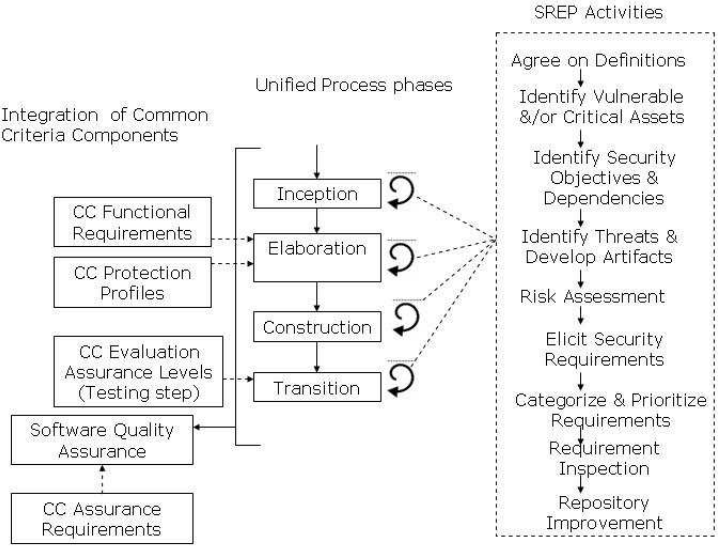

Fig. 1. SREP Overview

use of a security resources repository to support reuse of security requirements, assets, threats and countermeasures. The focus of this methodology seeks to build security concepts at the early phases of the development lifecycle.

As it is described in Fig. 1, the UP lifecycle is divided into a sequence of phases, and each phase may include many iterations. Each iteration is like a mini-project and it may contain all the core workflows (requirements, analysis, design, implementation, and test), but with different emphasis depending on where the iteration is in the lifecycle. Moreover, the core of SREP is a micro-process, made up of nine activities which are repeatedly performed at each iteration throughout the iterative and incremental development, but also with different emphasis depending on what phase of the lifecycle the iteration is in. Thus, the model chosen for SREP is iterative and incremental, and the security requirements and their associated security elements (threats, security objectives, etc.) evolve along 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 phase of the lifecycle and the activity of SREP, although the Software Quality Assurance (SQA) activities are per-formed along all the phases of the software development lifecycle, and it is in these SQA activities where the most of CC Assurance Requirements might be incorporated into.

2.1 The Security Resources Repository

The purpose of development with requirements reuse is to 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

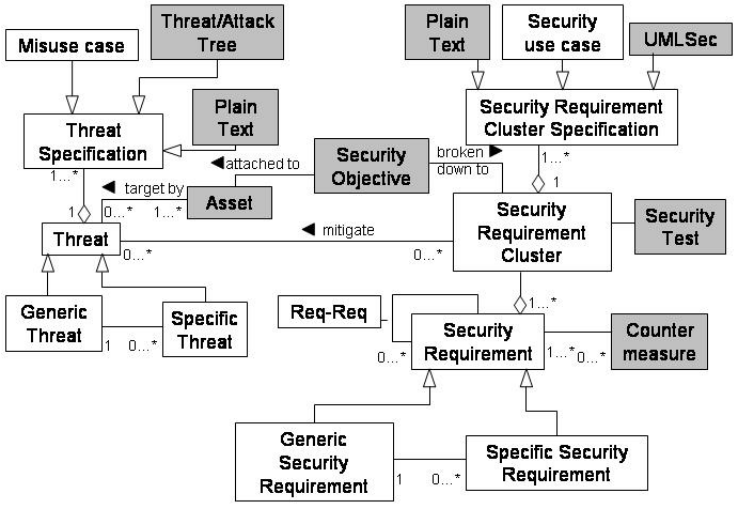

Fig. 2. Meta-model for security resources repository

us the fastest possible development cycles based on proven solutions. Therefore, we propose a Security Resources Repository (SRR), which stores all the security reusable elements. 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 in Fig. 2. The dark background in the objects represents our contribution to the meta-model.

As presented in Fig. 2, 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' are described 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.

It is known that an important part of the security of an IS can be often achieved through administrative measures, however the CC does not provide us with methodological support, nor contain security evaluation criteria pertaining

to administrative security measures not directly related to the IS security measures. Therefore, according to ISO/IEC 17799:2005 [7], 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 require-ments into software and system requirements format, these requirements along with the CC security requirements would be the initial subset of security requirements of the SRR.

3 Case Study

The case study we presented here is a representative case of a security critical IS in which security requirements have to be correctly treated in order to achieve a robust IS. It will be analysed the case of an administrative unit of the National Social Security Institute (of Spain), which has the porpoise of providing citizens e-government services. Here it will be studied the case of an e-government service which consists of an application (called Pension-App) that basically allows to provide information about the pension/s of a concrete citizen. Taking into account the constraint of space, this case study is unrealistically simple to enable points of SREP to be easily illustrated in this paper.

PensionApp is an application that allows citizens to obtain an official document which reflects the current amount and the status of their pension/s (whether it is being processed and the stage where it is at the moment of the request, or whether it has been successfully granted or rejected), it also allows citizens to update some personal data, such as their address and bank account number. One of the main design goals was maximum ease of use. Thus, citizens have online access to PensionApp through the Internet or they can go to an office of the National Social Security Institute, where a civil servant will provide them with an official paper document with the information requested about their pension or he/she will update the personal information of the citizens by interacting with other application which has been al-ready developed. Thus, a citizen can only obtain information about his/her own pension and update his/her personal information whereas a civil servant can get pension information and update personal information for a specified social security number of a person by interacting with the IS but through other application different from PensionApp. Thereby, we assume that initial functional requirements have been elicited and that there is only two functional requirements:

- Req 1: On request-1 from an EndUser, the system shall display in-formation about his/her pension. This request shall include the social security number of the EndUser.
- Req 2: On request-2 from an EndUser, the system shall update the personal information of the pensioner. This request shall include the social security number of the EndUser and changed personal data.

In addition we assume that the Organization has already introduced some elements into the Security Resources Repository (SRR), such as 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 requirements along with the CC security requirements will be the initial subset of security requirements of the SRR, which together with their associated security-elements (security objectives, assets, threats,...) will be the initial subset of security elements of the SRR.

SREP defines nine activities to be carried out as well as several iterations through the software development lifecycle, and each iteration will generate internal or external releases of various artefacts which altogether constitute a baseline, although in the following subsection of this paper we will only describe one iteration at the early stages of the software development lifecycle.

3.1 Activity 1: Agree on Definitions

In this activity we have to agree upon a common set of security definitions, along with the definition of the organizational security policies and the security vision of the IS. The following is a minute subset of the definitions that should be agreed.

- Information security: preservation of confidentiality, integrity and availability of information; in addition, other properties, such as authenticity, accountability, non-repudiation and reliability can be also involved [ISO/IEC 17799:2005].
- Threat: a potential cause of an unwanted incident, which may result in harm to a system or organization [ISO/IEC 13335-1:2004] [6].
- Availability: the property of being accessible and usable upon de-mand by an authorized entity [ISO/IEC 13335-1:2004].
- Confidentiality: the property that information is not made available or disclosed to unauthorized individuals, entities, or processes [ISO/IEC 13335-1:2004].
- Integrity: the property of safeguarding the accuracy and complete-ness of assets [ISO/IEC 13335-1:2004].
- Asset: anything that has value to the organization [ISO/IEC 13335-1:2004].

Then, the *Security Vision Document* will be written, in which it will be outlined the security vision of the IS. In this case, it will state that the most important asset is information, so from the security point of view it is important that confidentiality, availability and integrity of information, as well as authenticity, accountability, non-repudiation of the users and services are ensured.

3.2 Activity 2: Identify Vulnerable and/or Critical Assets

We have to perform an examination of functional requirements (Req1) (because according to CC assurance requirement ADV_FSP.3.1D the developer shall provide a functional specification) and we have realized that there is only one relevant asset type: Information. Other assets would need to be considered in a real

case study, including tangible assets such as money or products and intangible assets such as reputation. We can consider different types of Information:

- Personal information about the pensioner: name, social security number, address.
- Personal information about the pension/s: kind of pension (old-age/disability (type of disability)/widow's pension), amount of money, bank account number.

3.3 Activity 3: Identify Security Objectives and Dependencies

In this activity the SRR can be used, so that if the type of assets identified in the previous activity are in the SRR we will be able to retrieve their associated security objectives (SO). Otherwise we will determine the security objectives for each asset and we will take into account the security policy of the Organization as well as legal requirements and constraints in Spain and in the National Social Security Institute. We can identify the following security objectives:

- SO1: Prevent unauthorised disclosure of information. (Confidential-ity). Valuation - High.
- SO2: Prevent unauthorised alteration of information. (Integrity). Valuation - High.
- SO3: Ensure availability of information to the authorised users. Valuation - Medium.
- SO4: Ensure authenticity of users. Valuation - High.
- SO5: Ensure accountability. Valuation - Medium.

This is not a complete list, it should be refined in subsequent iterations (for example by establishing probability and dependencies between the security objectives), but it will be enough for this discussion. These security objectives will be written down in the *Security Objectives* Document with the help of the CC assurance classes (CC class ASE).

3.4 Activity 4: Identify Threats and Develop Artefacts

If the assets identified in the previous activity are in the SRR we will be able to retrieve their associated threats. Otherwise we will find all threats that can prevent the security goal from being achieved by instantiating the business use cases into misuse cases or by instantiating the threat-attack trees associated with the business and application pattern. In addition, we will analyse predefined threat lists for the type of assets selected and following the CC assurance requirement AVA_VAN.5.2E we will search in public domain sources to identify potential vulnerabilities in the IS. In Fig. 3 we present an example of misuse case diagram along with the possible attackers (crackers, thieves, etc.).

Therefore we identify several possible types of threats to Information:

- Generic Threat 1: Unauthorised disclosure of information.
- Generic Threat 2: Unauthorised alteration of information.

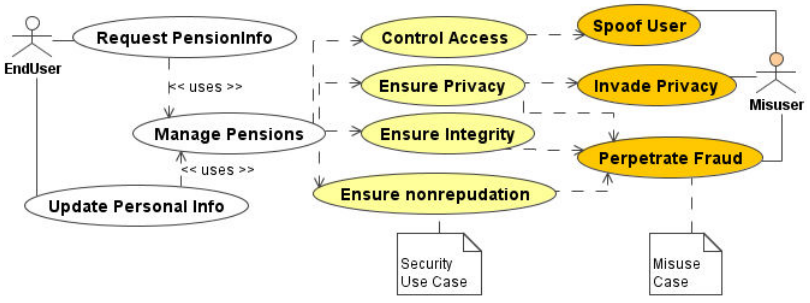

Fig. 3. Security Use Cases and Misuse Cases

- Generic Threat 3: Unauthorised unavailability to information.
- Generic Threat 4: Spoof user.

Then, we will develop the Generic Threats together with the Specific Threats (which are several paths of these Generic Threats) if there are no threats in the SRR that match with the previous identified types of threats. So we will present an example of a specification of a *Generic Threat*(Generic Threat 2) and a *Specific* Threat using misuse cases as a method of specification in Tables 1 and 2.

Finally, with the former information we have achieved in this activity, we will constitute the first version of the *Security Problem Definition Document* with the help of the CC assurance classes (CC class ASE). As it is in this document where the assumptions are written down, we would like to reflect the fact that we do not take into consideration, because it is not the main object of this specific work, possible attacks on the provider and consumer organizations, on the network infrastructures or on the infrastructure in use, along with other elements at an organizational level (and not only system-level elements).

3.5 Activity 5: Risk Assessment

Having identified the threats, we shall now go on to determine the probability of each threat and to assess its impact and risk. In order to carry out this task, we will use a technique proposed by the guide of techniques of MAGERIT [11] and which is based on tables to analyse impact and risk of threats.

For the time being we are going to evaluate risk and impact with five possible values: Very Low, Low, Medium, High and Very High. The likelihood of a threat could be: Very Frequent (daily event), Frequent (monthly event), Normal Frequency (once a year), Rarely (once in several years). We have therefore to produce a table of threats, attacks (misuse cases: MUC) and risks to register the evaluation of impact and risk regarding the threats we have identified. In Table 3 we will present an example of the analysis of the risk of one threat previously detailed in the former activity. All of this is captured in the *Risk Assessment Document* which will be also refined in subsequence iterations.

Table 1. Generic Threat Specification using misuse cases (GMUC)

Name of Misuse Case: Attack on the content of a HTTP message [message name] of the User		
ID: GMUC-2-2-1-1 [GMUC-Security Objective-Generic Threat- Iteration- GenericMisUseCase]		
PROBABILITY: [Very Frequent | Frequent | Normal Frequency | Rarely]		
Summary: The attacker type [attacker type] gains access to the message [message | interaction] [name] exchanged by the [consumer | provider] agent [agent name] and the [consumer | provider] agent [agent name] and [modifies | deletes | inserts [part/s]] of the message at the [transport | http]-level situated in the [header | body | attachment] with the object of [objective].		
Preconditions: 1) The attacker has physical access to the message. 2) The attacker has clear knowledge of the structure and meaning of the message.		
User Interactions	**Misuser Interactions**	**System Interactions**
The User sends a message [name of message]		
	The attacker [type of attacker] [name of attacker] intercepts it and identifies the part of the message to modify and [deletes | replaces | add] information and he/she forwards it on to the System Agent	
		The System Agent receives the corrupted message and processes it wrongly due to the altered semantic content
Postconditions: 1) The system will remain in a state of error with respect to the original intentions of the User agent [name of user agent]. 2) In the register of the system in which the Provider Agent [name of provider agent] was executed the request received with an altered semantic content will be reflected.		

Table 2. Specific Threat specification

Name of Misuse Case: Attack on the content of the http-message UpdatePersonalInfo from End-User to PensionApp		
ID: *SMUC-2-2-1-1-1* [SMUC-Security Objective-Generic Threat- Iteration-GenericMisUseCase-SpecificMisUseCase]		
PROBABILITY: *FREQUENT*		
Summary: The external attacker type gains access to the UpdatePersonalInfo message exchanged between the consumer agent (browser of the End-User) and PensionApp, and modifies the part of the HTTP message that contains the pensioner's bank account number with the intention of changing its meaning by modifying the account number to fit one owned by the attacker		
Preconditions: 1) The external attacker has physical access to the message. 2) The external attacker has clear knowledge of where within the UpdatePersonalInfo message is located the account number.		
User Interactions	**Misuser Interactions**	**System Interactions**
The User Agent sends an UpdatePersonalInfo message		
	The external attacker intercepts it and identifies the part of the message to modify the bank account number and he/she forwards it on to PensionApp	
		PensionApp receives the corrupted UpdatePersonalInfo message and processes it wrongly due to its altered semantic content. That is, it establishes that the consumer agent wishes as new bank account number the account number modified by the attacker.
Postconditions: 1) PensionApp will remain in a state of error with regard to the original intentions of the End-User. 2) In the register of the system in which PensionApp was executed, the request received with an altered semantic content will be reflected		

Table 3. Table of Threats, Attacks and Risks

Table of Threats, Attacks and Risks - Iteration 1				
Threat	Impact	Attack	Probability	Risk
1.2.1.1.1.1 Alteration of the information	LOW, if there is not pension information modified	*SMUC-2-2-1-1-1*	HIGH	LOW
	HIGH if the opposite is the case.	*SMUC-2-2-1-1-1*	HIGH	HIGH

3.6 Activity 6: Elicit Security Requirements

In order to derive security requirements, each security objective is analysed for possible relevance together with its threats which imply more risk, so that 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 are selected. First of all, we will use domain knowledge to transform the entities described in the security objectives into entities in the functional requirement. In this case, it is straightforward, the security objectives refer to information and we know that it is pension information or pensioner information in the context of the functional requirements.

Then, we will transform the security objectives (Confidentiality, Integrity, Availability, Authenticity, Accountability) into constraints on the operations that are used in functional requirements. Additionally, we will search in the CC security functional requirements catalogue (which has been previously introduced together with the CC assurance requirements into the SRR) security requirements which mitigate the threats that can prevent the security objective from being achieved, therefore in this case, we will search for ensuring the integrity, availability, authenticity and accountability of PensionApp. Moreover, we will search in the CC security assurance requirements catalogue to determine the assurance requirements which ensure the secure development of the IS.

The security requirements (SR) that we identify are the following ones:

- SR1: The security functions of PensionApp shall use *cryptography* [assignment: *cryptographic algorithm* and *key sizes*] to protect confidentiality of pension information provided by PensionApp to an EndUser. (CC requirement FCO_CED.1.1)
- SR2: The security functions of PensionApp shall identify and authenticate an EndUser by using *credentials* [assignment: *challenge-response technique based on exchange of encrypted random nonces, public key certificate*] before an EndUser can bind to the shell of PensionApp. (CC requirements FIA_UID.2.1 & FIA_UAU.1.1)
- SR3: When PensionApp transmits pension or pensioner's information to EndUser, the security functions of PensionApp shall provide that user with the *means* [assignment: *digital signature*] to detect [selection: modification, deletion, insertion, replay, other integrity] anomalies. (CC requirement FCO_IED.1.1)
- SR4: The security functions of PensionApp shall ensure the availability of the information provided by PensionApp to an EndUser within [assignment:

a defined availability metric] given the following conditions [assignment: conditions to ensure availability]. (CC requirement FCO_AED.1.1)
- SR5: The security functions of PensionApp shall require evidence that PensionApp has pension information to an EndUser and he/she has received the information. (CC requirement FCO_NRE.1.1)
- SR6: The security functions of PensionApp shall store an audit record of the following events [selection: *the request for pension information, the response of PensionApp*] and each audit record shall record the following information: date and time of the event, [selection: *success, failure*] of the event, and EndUser identity. (CC requirements FAU_GEN)

Due to the former security requirements the first functional requirements (Req1 and Req2) have to be updated, so that they will be as follows:

- Req 1': On request from an EndUser, the system shall display information about his/her pension. This request shall include the social se-curity number of the EndUser and the EndUser's Credentials.
- Req 2': On request-2 from an EndUser, the system shall update the personal information of the pensioner. This request shall include the social security number of the EndUser and the EndUser's Credentials and the changed personal dates.

In Table 4 we will present an example of a *Generic Security Requirement specification* using security use cases as a method of specification. Finally, the *Security Requirements Specification Document* is written in this activity and it will be refined in subsequence iterations because we try to avoid unnecessarily and prematurely architectural/design mechanisms specification.

3.7 Activity 7: Categorize and Prioritize Requirements

According to the impact and the likelihood of the threats, that is according to the risk, we will rank the security requirements as follows: 1- SR1; 2- SR2; 3- SR3; 4- SR5 and SR6; 5- SR4.

3.8 Activity 8: Requirements Inspection

In this activity, we will generate the *Validation Report*, thereby we will review the quality of the previous work with the help of the CC assurance requirements, these assurance requirements will result from the determined EAL, which was agreed with the stakeholders in the first activity, although it could be modified in subsequent iterations. Supposing we agreed EAL1 (functionally tested) the assurance components that we will use will be presented in the Table 5.

Then we will write the first version of the *Security Requirements Rationale Document* with the help of the CC assurance classes (CC class ASE), showing that if all security requirements are satisfied and all security objectives are achieved, the security problem defined previously is solved: all the threats are countered, the organizational security polices are enforced and all assumptions are upheld.

Table 4. Generic Security Requirement Specification

Name of the Generic Security Use Case: ***Ensure the integrity of a HTTP message [name of the message]***				
ID: GSUC-2-2-1-SR3-1 (is the first GSUC associated with the GMUC-1-1-1) [GSUC-Security Objective-Generic Threat- Iteration-SecuritvRequirement- GenericsSecurityUseCase]				
Preconditions: 1) The attacker [attacker type] [name of attacker] has physical access to the message. 2) The attacker [attacker type] [name of attacker] has clear knowledge of the structure and meaning of the message				
Misuser Interactions	**System Requirements**			
	Interactions of the User Agent	**Actions of the User Agent**	**System Interactions**	**System Actions**
	The User Agent [name of agent] builds a private http message [name of message] and sends it to the System	The User Agent [name of agent] should try to ensure that the modifications that may occur in the message [name of the message] as it is convoyed will be detected in an obvious way by the System		
The attacker [type of attacker] [name of attacker] intercepts it and identifies the part of the message to modify and [deletes | replaces | add] information and he/she forwards it on to the System Agent				
			The System Agent receives the altered message [name of message]	The System Agent detects that the message [name of message] was altered in transit, rejects it and executes [operations]
Postconditions: 1) The System Agent will have executed [operations] [name of agent] with the aim of detecting that the message was altered in transit.				

Table 5. EAL 1 Common Criteria Assurance classes and components

Assurance Class	Assurance components
ADV: Development	ADV_FSP.1 Basic functional specification
AGD: Guidance documents	AGD_OPE.1 Operational user guidance
	AGD_PRE.1 Preparative procedures
ALC: Life-cycle support	ALC_CMC.1 Labelling of the TOE
	ALC_CMS.1 TOE CM coverage
ASE: Security Target evaluation	ASE_CCL.1 Conformance claims
	ASE_ECD.1 Extended components definition
	AE_INT.1 ST introduction
	ASE_OBJ.1 Security objectives for the operational environment
	ASE_REQ.1 Stated security requirements
	ASE_TSS.1 TOE summary specification
ATE: Tests	ATE_IND.1 Independent testing – conformance
AVA: Vulnerability assessment	AVA_VAN.1 Vulnerability survey

3.9 Activity 9: Repository Improvement

We will store in the SRR the new elements, in this case in this iteration the Generic and Specific Threats and Requirements which were developed in the activities 4 and 6 will be stored. After all, we will write the *Security Target document* of the CC. This activity will be performed coinciding with the milestone at the end of each phase of the UP.

4 Lessons Learned

Among the most important lessons learned we may stand out from the case study presented above we can highlight the following ones:

- The application of this case study has allowed us to improve and refine the following activities of SREP: identification of security objectives, identification of threats and elicitation of requirements.
- Tool support is critical for the practical application of this process in large-scale software systems due to the number of handled artefacts and the several iterations that have to be carried out.
- As it is an iterative and incremental security requirements engineering process, we have realized that this philosophy lets us take into account changing requirements, facilitates reuse and correct errors over several iterations, risks are discovered and mitigated earlier, and the process itself can be improved and refined along the way.
- Regarding to the experience with the Common Criteria, we have realized that it is sometimes difficult to find the right meaning of the CC requirements, it would be easier if the CC provides examples for each security requirement. However the CC provide us with an important help in order to treat security requirements in a systematic way, in spite of the fact that CC requirements have complex dependencies and the CC does not provide us with any method/guide to include them into the software development process, so that a modification in one document often leads to modify several other documents.

5 Conclusions

In our present so-called Information Society the development of more and more sophisticated approaches to ensuring the security of information is becoming a need. In this paper we demonstrate how the security requirements for a security critical IS can be obtained in a guided and systematic way by applying SREP. Starting from the concept of iterative software construction, we propose a microprocess for the security requirements analysis, made up of nine activities, which are repeatedly performed at each iteration throughout the iterative and incremental development, but with different emphasis depending on where the iteration is in the lifecycle. Therefore the contribution of this work is that of

providing a standard-based process that deals with the security requirements at the early stages of software development in a systematic and intuitive way, which is based on the reuse of security requirements, by providing a Security Resources Repository (SRR), together with the integration of the Common Criteria (ISO/IEC 15408) into software development lifecycle. Moreover, it also conforms to ISO/IEC 17799:2005 with regard to security requirements (sections: 0.3, 0.4, 0.6 and 12.1). Hence, it is a very helpful process for security critical Information Systems. 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 more real case studies in order to complete and detail more SREP.

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 the FEDER and the CALIPO (TIC2003-07804-CO5-03) and the 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. Booch, G., Rumbaugh, J., and Jacobson, I. "The Unified Software Development Process". *ed. Addison-Wesley*, 1999.
3. Breu, R., Burger, K., Hafner, M., and Popp, G. "Towards a Systematic Development of Secure Systems". *Proceedings WOSIS 2004*, 2004: p. 1-12.
4. Firesmith, D.G. "Engineering Security Requirements". *Journal of Object Technology*, 2003. 2(1): p. 53-68.
5. Firesmith, D.G. "Security Use Cases". *Journal of Object Technology*, 2003. p. 53-64.
6. ISO/IEC_JTC1/SC27. "Information technology - Security techniques - Management of information and communications technology security - Part 1: Concepts and models for information and communications technology security management". *ISO/IEC 13335*, 2004.
7. ISO/IEC_JTC1/SC27. "Information technology - Security techniques - Code of practice for information security management". *ISO/IEC 17799*, 2005.
8. ISO/IEC_JTC1/SC27. "Information technology - Security techniques - Evaluation criteria for IT security". *ISO/IEC 15408:2005 (Common Criteria v3.0)*, 2005.
9. Kim., H.-K. "Automatic Translation Form Requirements Model into Use Cases Modeling on UML". *ICCSA 2005, LNCS*, 2005: p. 769-777.
10. Kotonya, G. and Sommerville, I. "Requirements Engineering Process and Techniques". *Hardcover ed*, 1998. 294.
11. MAP. "Metodología de Análisis y Gestión de Riesgos de los Sistemas de Información (MAGERIT - v 2)". *Ministry for Public Administration of Spain*, 2005.
12. Massacci, F., Prest, M., and Zannone, N. "Using a security requirements engineering methodology in practice: The compliance with the Italian data protection legislation". *Computers Standards and Interfaces, 27*, 2005, p.445-455.

13. Dermott, J. and Fox, C."Using Abuse Case Models for Security Requirements Analysis". *Annual Computer Security Applications Conference*, Phoenix (AZ), 1999.
14. Mellado, D., Fernández-Medina, E., and Piattini, M."A Common Criteria Based Security Requirements Engineering Process for the Development of Secure Information Systems". *Computer Standards and Interfaces* , 2006.
15. Mellado, D., Fernández-Medina, E., and Piattini, M."A Comparative Study of Proposals for Establishing Security Requirements for the Development of Secure Information Systems". *The 2006 International Conference on Computational Science and its Applications (ICCSA 2006), Springer LNCS 3982* , 2006. 3: p. 1044-1053.
16. Mouratidis, H., Giorgini, P., Manson, G., and Philp, I."A Natural Extension of Tropos Methodology for Modelling Security". *Workshop on Agent-oriented methodologies, at OOPSLA 2002* ,Seattle (WA), 2003.
17. Popp, G., Jürjens, J., Wimmel, G., and Breu, R."Security-Critical System Development with Extended Use Cases". *10th Asia-Pacific Software Engineering Conference* , 2003, p. 478-487.
18. Sindre, G., Firesmith, D.G., and Opdahl, A.L."A Reuse-Based Approach to Determining Security Requirements". *9th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'03)* , Austria, 2003.
19. Toval, A., Nicolás, J., Moros, B., and García, F."Requirements Reuse for Improving Information Systems Security: A Practitioner's Approach". *Requirements Engineering Journal* , 2001, p. 205-219.
20. Walton, J.P."Developing a Enterprise Information Security Policy". *ACM Press: Proceedings of the 30th annual ACM SIGUCCS conference on User services.* , 2002.
21. Yu, E."Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering". *A3rd IEEE International Symposium on Requirements Engineering (RE'97)* , 1997, p. 226-235.