Title: Security in Requirement Engineering for Qualitative Products

Isma Hameed

***Abstract*—The objective of defining security requirements for a system is to map the practical security requirements statements with the results of risk and threat analysis. Under investigation the securities risks of the system can manage (cancel, mitigate or maintain). Like system engineering security engineers must develop appropriate threat models. Security engineers must select those security measures which are most needed for market success. To improve the quality of the artifacts such as requirements documents the software engineers must use models early in the life cycle. In our research we have proposed a model "integration of security with functional requirements model". By using this model the security issues defined at the requirement engineering phase and it has followed in the next stages of the SDLC, We had proved our results by checking the efficiency of the architecture after applying the proposed model on a payment system.**

***Index Terms*—Security, requirement engineering, quality.**

I. Introduction

In the security evaluation the security requirements play an important role. The requirements show the whole process of security conformation in the product. Security requirements are frequently missing in the requirements elicitation process, and tend to be ignored afterward. An organization can make sure that the resulting product successfully meets security requirements, if practical approaches to security requirements continue to be developed and mechanisms of security requirements are identified to support organizational use [7]. Is in present such a thing as a software system that doesn't need to be secure? Almost every software controlled system faces threats from possible adversaries and from Internet aware client applications running on Pc's. While still delivering to the customers, Software engineers must be conscious about these threats and engineer systems with reliable defenses [3].

In an increasingly complex environment the security is a multifaceted issue. A lack of security integration and understanding of the application development process creates an environment that is encouraging to promote security deficiencies [5]. Security is one of the areas that need to be careful as an essential part of the project. For example a development team would not start development without knowing the target platform for their project or the major interfaces their application needs to have with other application. If we don't include the security requirements from the initial phase of the project then there'll not be only increased in project costs, but also the time taken'll increase naturally [9] fulfill the requirements the security mechanisms are developed and to reduce vulnerabilities requirement engineers used the security requirements as constraints on functional requirements [8].

In effect when software engineers are dealing with software security, it is a common failure that they too much focus on the development phase of the SDLC, not enough on the other phases [1]. Security requirement engineering process is the most ignored part of the security enhanced software development lifecycle. The major reason for this mistake is that security is assumed to be a technical issue and as a result best handled during architectural and design phase or better still during implementation. While software requirements are frequently written by non technical business analysts. This is a common conclusion.

Most requirements engineers are not skilled at all in security. A small amount of people that have been skilled have only been given a general idea of security architectural mechanisms such as passwords and encryption rather than in actual security requirements [4].

II. Materials and Methods

The proposed model: (Integration of Security with functional requirements model):

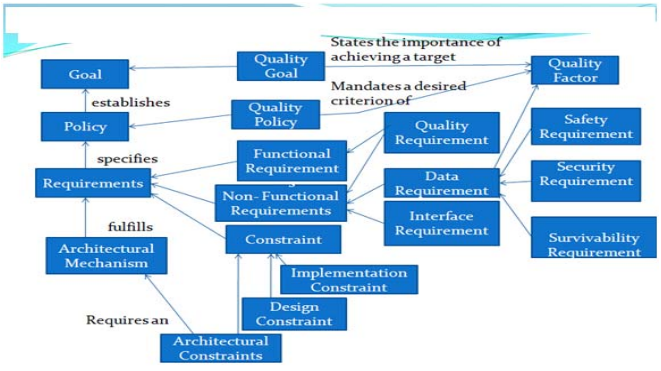

Fig. 1. Integration of security with functional requirements

Applying the integration of security with functional requirement model to a payment system

A. Identify Requirements:

Requirements of a system can be classified into functional and non-functional requirements here in our case security requirements are considered as a functional

Manuscript received September 20, 2012; revised November 5, 2012.

Isma Hameed is with Department of Computer Science University of Agriculture Faisalabad, Pakistan (e-mail: ms_isma222@hotmail.com).

requirement. The way in which a user uses a system can be described by using use case driven approach, that's why use case diagram is frequently used for capturing functional requirements.

B. *Identify actors and Use Cases of Payment System*

The actors we identified for the payment system are:

- Customer
- Cashier
- Manager
- Payment authorized service
- System administrator

C. *Identify Quality Attributes*

Quality attributes can be goals, constraints or assumptions of stakeholders. By analyzing the primary set of requirements, the possible quality attributes are identified, for example, if the cashier enters the invalid identifier and the system signals error and rejects entry then security is a concern that the system needs to address here in this system security is treated as functional requirement.

D. *Build the Use Case Diagram*

In one use case diagram the set of all use cases can be represented. In that use case we can see the existing relationships between use cases and actors. The use case diagram of the payment system is shown in the following figure.

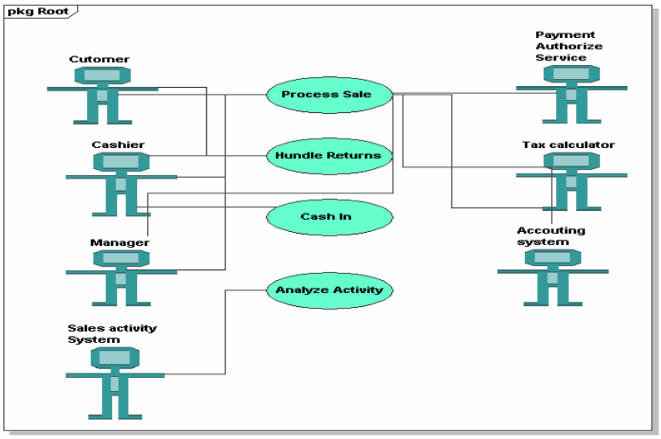

Fig. 2. The use case diagram of payment of sale

E. *Security Requirements in Payment System*

TABLE I: SECURITY REQUIREMENTS IN PAYMENT SYSTEM

	Implementation
Requirements	The system must : 1. Maintain secrecy and privacy: allowing read access to only those users who have been authorized. (Confidentiality) 2. Ensuring the completeness and accuracy of information. (Integrity)

F. *Integrate Security with Functional Requirements:*

The following diagram represents security requirements use cases. Security is festering into confidentiality and integrity. Integrity should be used after execution of the use case and confidentiality should be used before it.

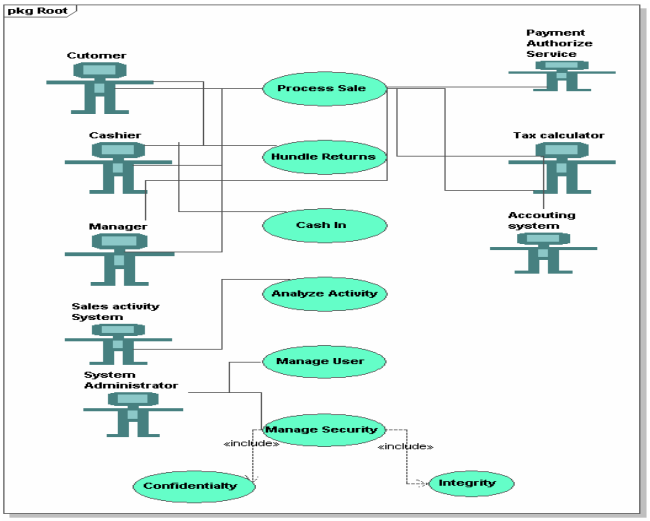

Fig. 3. Integrate security with functional requirements use case

III. RESULTS AND DISCUSSION

A number of research efforts have been addressed the issue of integrating security into the system development process. They have been focus on very different aspects for example from design of access control mechanisms to modeling of the behavior of the system, from the explanation of principles for conflict analysis and categorization of the description of security patterns [6]. Several requirement engineering research and practice have been addressed the capabilities that the system will provide. So from the user's point of view a lot of attention is given to the functionality of the system, but a little bit attention is given to what the system should not do [2]. This research was based on the same methodology as described above that the security issues must be defined at the requirement engineering level subsequently they can elicit, analyze and validate. In our payment system the security requirements were integrated with functional requirements.

Comparison of Payment system architecture is without security and with security.

TABLE II: ARCHITECTURE ASSESSMENT RESULTS

	Without security	With security
Average mean score on Architecture Assessment (out of 100)	55	75.5

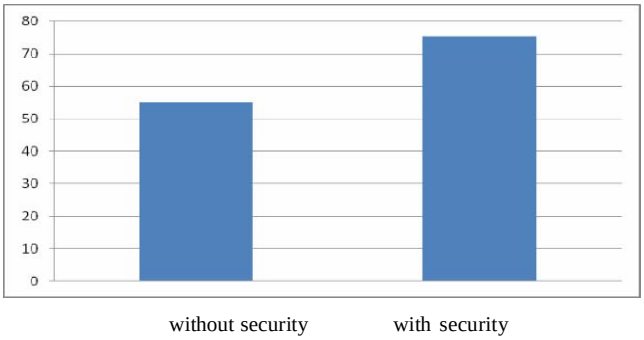

Fig. 4. Comparison of payment system architecture without security and with security

The Table II of assessment architecture results showed that the with security systems architectures were more efficient as compared to without security systems architectures.

Comparison of payment system architectures with different security levels: Security is festering into confidentiality and integrity. Confidentiality should be used before the execution of the use cases and integrity should be used after it. In the following table the architecture assessments have done with levels of security, First without security, second security with confidentiality, third security with integrity and then security with confidentiality and integrity.

TABLE III: ARCHITECTURE ASSESSMENT RESULTS WITH DIFFERENT SECURITY LEVELS

	Without security	Security with confidentiality	Security with integrity	With Security
Average mean score on Architecture Assessment (out of 100)	55	65.2	69	75.5

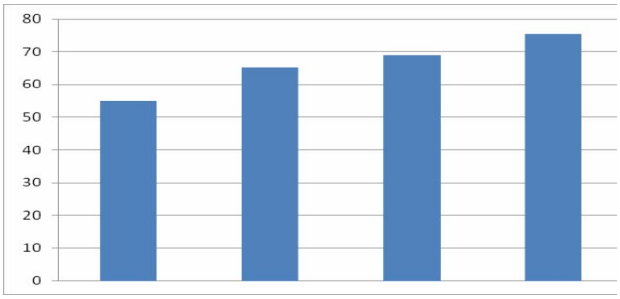

Fig. 5. Comparison of payment system architecture with different security levels

The table III and Fig. 5 shows that as the levels of security putting in to system the architecture of that system were becoming the more efficient because that system were covered all aspects of threats.

REFERENCES

[1] R. Araujo, Security Requirements Engineering: A Road Map. Software Magazine – Security Requirements Engineering A Road Map.htm. 2007.

[2] M. Bishop, "Computer Security: Art and Science. Boston," *MA: Addison-Wesley Professional*, 2002.

[3] P. T. Devanbu and S. Stubblebine, Software Engineering for Security: a Roadmap. Department of Computer science, University of California USA. 2001.

[4] D. G. Firesmith (2003), Common Concepts Underlying Safety, Security, and Survivability Engineering, Technical NoteCMU/SEI-2003-TN-033.

[5] W. B. Gilsson, A. McDonald, and R. welland, Web Engineering Security: A practitioner's Perspective, ACM 1-59593-352-2/06/0007. 2007.

[6] F. Massacci and N. Zannone, Detecting conflicts between functional and security requirements with secure tropos: John Rusnak and the allied Irish bank, University of Toronto. 2007.

[7] N. R. Mead, Security requirements engineering, Software engineering Institute Carnegie Mellon University, 2006.

[8] Savola, Towards Security Evaluation Based on Evidence Collection, VTT Technical Research Centre of Finland, 2006.

[9] S. Wing, The importance of incorporating security requirements within system architecture rather than incorporating retro fitting controls to an insecure design, 2006.