Scientific Research Publishing

Journal of Software Engineering and Applications, 2017, 10, 873-883
http://www.scirp.org/journal/jsea
ISSN Online: 1945-3124
ISSN Print: 1945-3116

Design of Secure and Traceable Requirement Engineering Process for Security-Sensitive Projects

Abdul Ahad[1], Luqman Tariq[2], Saba Niaz[1], Muhammad Inam[3]

[1]Department of Computer Science, Virtual University, Lahore, Pakistan
[2]Department of Computer Science, Preston University, Islamabad, Pakistan
[3]Department of Computer Science, Beijing University of Technology, Beijing, China
Email: ahad9388@gmail.com

How to cite this paper: Ahad, A., Tariq, L., Niaz, S. and Inam, M. (2017) Design of Secure and Traceable Requirement Engineering Process for Security-Sensitive Projects. *Journal of Software Engineering and Applications*, **10**, 873-883.
https://doi.org/10.4236/jsea.2017.1012049

Received: September 19, 2017
Accepted: November 10, 2017
Published: November 13, 2017

Abstract

With continuous evolution in software industry, security is becoming very important in software projects. However, in many development methodologies, security is thought to be added in the project at later stages of the development lifecycle. There are also many proposed methodologies where the security measures are considered at requirement engineering stage of the development lifecycle, but many of them still do not seem adequate for applicability due to the reason that these approaches do not provide sufficient support for mapping the security requirements to the later stages of development. So, we are in need of a software requirement engineering approach, which is not only helpful in security requirement specification at requirement engineering stage but also provides support for using the specified security requirements at later stages of development. To meet this requirement, we introduce a new method Secure and Traceable Requirement Engineering Process (STREP). This method also helps the non-security-expert requirement engineers to specify requirements in such a way that the specified requirements can be used to derive security related test cases. STREP method not only deals with security issues of the system at requirement engineering stage, but also makes the security requirements more traceable to be used at later stages of development lifecycle, and as a result, secure systems are produced that are also usable as the customer wishes.

Keywords

Traceable, Usable, Security Requirements, Requirement Specification, Development Lifecycle, SRE Method

1. Introduction

In a software development lifecycle, this requirements engineering is the first important and main stage. It is the requirements engineering stage where software developers and the users/customers meet each other to decide what functionalities the software to-be-developed should have [1]. Requirement engineering stage has strong effects on the software development lifecycle. Good requirement engineering helps in producing successful projects. Bad requirement engineering introduces errors in the requirements specification, and these errors are propagated to next stages of software development lifecycle, resulting in an unsuccessful system development. Security related requirements and non-functional requirements can get a lot befit from a good requirement engineering process.

Security-sensitive systems require the safety of their resources and information. Security is essential all the way from processing and storing the resources and information to transmitting them to other remote systems. There may be many threats to security of resources and information e.g. unauthorized access to resources and information, changing the information during the transmission, or disabling the authorized access [2] etc.

Security specific requirements are of the great importance for security sensitive systems at the requirements specification stage, and what is more importance is to provide some means to shift these security specific requirements to next stages of the development lifecycle. Our major goal is to introduce a security specific requirement engineering process that is not only helpful at requirements specification stage but also helpful at the next stages of development lifecycle. The requirement engineering process should be applicable at system architecture, system design, system implementation, system testing, and system deployment stages of the development lifecycle. In the development of security-sensitive system, secure requirements are used at all stages of development lifecycles, and therefore, there should be proper processes for specifying secure requirements at requirement engineering stage. The unhappy fact is that the current techniques that are used to collect secure requirements [3] are not very much beneficial at later stages of the development lifecycle.

In this paper, a process STREP, Secure and Traceable Requirement Engineering Process, is being proposed with the intent to be helpful and Traceable at the later stages of the development lifecycle. The basic purposes of proposing the STREP method are:

1) To have a more efficient and user-friendly process for security requirement specification. To achieve this, the benefits of three security requirement engineering techniques have been combines. These techniques are Misuse Cases [4], USeR, and CLASP [5]. These will help us in specifying clearer, more complete, and better understandable security requirements.

2) To specify security requirements which have more traceability features and can be used at all the later stages of development lifecycle. Our proposed approach will do it for us.

3) To provide guidelines and some tool assistance that will help in mapping the securing requirements to security related test cases [6]. STREP will help us in this mapping.

Organization of paper as follows: In Section II, discussion about software design, design concept, design attribute, design process and design principles. In Section III, the target problem will be discussed and previous work on the problem will be briefly described. In Section IV, reasons for proposing the STREP method will be described. In Section V, the STREP method will be explained. In Section VI, the expected contribution of the proposed method will be discussed. Section VII will conclude the paper.

2. Software Design

Software design is a process of applying software solutions to one or more groups of problems. Software requirements analysis (SRA) is one of the main components of software design. Software requirements analysis (SRA) is the part of software development process which lists specifications used in the software engineering. If the software is user centred or semi-automated, software design may include user experience design yielding a storyboard to help show those specifications. If the software is completely automated, a software design may be as simple as text or flow chart to describing a planned sequence of events.

2.1. Design Concept

Design concept provided software designer with a groundwork from which more classy methods can be applied. A set of fundamental design concepts has evolved. They are following

1) *Abstraction*: Abstraction is a process or result of generalization by minimizing the information content of a concept or a noticeable phenomenon, typically in order to recall the only information which is applicable for specific purpose. It is a performance of representing important features without including the background explanations.

2) *Refinement*: Refinement is the process of explanation. Hierarchy is developed by decaying a macroscopic statement of a function in step-wise manner till programming language statements are reached. In every step, one or more commands of a given program are decayed into many detailed instructions. Refinement and Abstraction are complementary concepts.

3) *Modularity:* Software architecture is separated into components which is called modules.

4) *Software Architecture*: Software Architecture is the overall structure of software and this structure provides theoretical integrity for a system. A good software architecture will produce a good return on investment with respect to the wanted outcome from the project, e.g. in terms of quality, cost, schedule and performance.

5) *Control Hierarchy*: A structure of a program which represents the organi-

zation of program elements and implies a hierarchy of control.

6) *Structural Partitioning*: A program structure can be divided into both horizontal and vertical. Horizontal partitions describe the separate branches of a modular hierarchy for every major program function. Vertical partitions suggest that the control and work should be distributed top down in program structure.

7) *Data Structure:* Data Structure represents the logical relationship between the individual elements of data.

8) *Software Procedure:* Software Procedure focuses on the processing of every module individually.

9) *Information Hiding:* Modules should be designed and specified so that the information inside module is inaccessible to every other module that has no need for such information.

2.2. Design Process

Design process is a repetitive process through which requirements are translated into a model. The design is represented at a high level of abstraction, *i.e.* a level to design representation at much lower levels of abstraction.

Design is only way by which we accurately a finished software product or system.

The design process must follow the following three general points.

1) The design must implement all explicit requirements contained in analysis model and it must accommodate all implicit requirements desired by the customers.

2) The design must be readable; understandable for those who generate code and for those who test and subsequently maintain the software.

3) The design should provide a complete picture of the software. Its data structure, function and behavior etc.

Each of the above maintained characteristics represents the goal of a good design. The designer can achieve this goal by following these characteristics. To evaluate the quality of design representation, we have to establish technical criteria for a good design. These are

1) A design should be a modular.

2) A design should contain both data and procedural abstraction.

3) A design should be derived using repeatable method.

4) A design should lead to interfaces.

5) A design should display a hierarchal.

The above mentioned technical criteria can be achieved by applying fundamental design principles, systematic, methodologies throw review etc.

2.3. Software Design Quality Attributes

The goal of design process is not simply to process a design for a system, instead of the goal is to find the best possible design. Within the limitation imposed by requirements and physical and social environments in which the system will operate.

Some quality attributes for a software system design are following [7]

1) *Verifiability:* The first objective of design is verifiability which means that the design should be verifiable and correct. It refers to set of activities that ensure the software correctly implemented a specific function.

2) *Completeness:* It means that design should be complete in all aspects. All the number of modules needed should be specified.

3) *Consistency:* The design should be consistence, *i.e.* one module should not have two different responses to different entities.

4) *Efficiency:* The design should be efficient; an efficient system is that which consumes processor time and requires less memory.

5) *Traceability:* It is an important property that can aid design verification. It requires that all desire elements must be traceable to the requirements.

6) *Simplicity and understand ability:* The design must be readable; understandable guide for those who generate code and for those who test and subsequently maintain the software.

2.4. Design Principles

The design of a large system is a complex task. The basic guiding principles are followed to produce a good design of a system. These design principles reduce the complexity as well as the efforts needed for design and design cost. The error that can be occurred during design process also reduced this way.

The basic design principles help a software engineer to navigate the design process. The basic design principles suggested by the Davis for the software engineering are given below.

1) The design process should not suffer from Turmel sight. It means a good designer should be considering alternative approaches; judging each based on the requirements of problems. 1.

2) The resources available to do the job and the design concepts.

3) The design should be traceable to the analysis model.

4) The design should not re-invent the wheel. The system is constructed by dividing the problem into manageable small pieces of pattern that can be solved separately. These patterns are referred as re-usable design components. They should be chosen as an alternative to re-invent.

5) The design should minimize the intitulé between the software and problem as it exists in the real world.

6) The design should display uniforms and integration. It means that rules and format should be defined for a design team before design work begins.

7) Design should to be structured well to accommodate new changes.

8) The design should be structured to be acceptable even when varying data, events or operating conditions are encountered.

9) The design is not coding and coding is not design.

10) The design should be assessed for quality as it is being created. For this purpose, various design concepts and design measures are available for the help of designer in assessing quality.

11) The design should be reviewed to minimize the conceptual errors.

When the above-mentioned design principles are properly applied, the system design is created that shows both external and internal quality factors.

1) The external factors are those properties of the system or software that can be observed by the users.

2) The internal factors are those properties of the software can be observed by the software engineer. The internal factors lead to the high-quality design.

3. Problem Definition and Related Work

Security needs become important when stakeholders find that the system to-be-developed has some assets (e.g. money or confidential information) of great value for the organization, and stakeholder want to save them from attacks or damage [8] [9]. Software requirement engineering process should, therefore, be concerned about saving these valuable assets from requirement perspective. Different approaches are available for gathering secure requirements, but unfortunately, they do not provide significant support for using these secure requirements at the later stages of the software development lifecycle. We believe that security requirements need equal attention at the later stages of development lifecycle as well so that the traceability and usability of the secure requirement may be ensured at later stages [10].

In [3], a survey has been conducted to know the support of existing SRE methods for security requirements. The support was evaluated against five areas of security requirements. These five areas were requirement elicitation, requirement analysis, requirement specification, requirement maintenance, and requirement support for later stages of the development. SRE methods were selected on the basis of their maturity towards industry and academia. References for each of these SRE methods have also been given in [3]. Selected SRE methods were CLASP [5], Misuse Cases, USeR method, abuser stories, secure TROPOS, SQUARE [11], Anti-Models, and security Problems Frames etc.

Survey revealed that these methods have shortcomings in three different important areas.

1) In most of the methods, it is difficult to specify the security requirements, and if specified, it is difficult to understand the security requirements.

2) These methods provide little support for using the security requirements at later stages of development lifecycle.

3) These methods do not provide sufficient support for non-security experts to get benefit from them.

In this paper, we are proposing "Secure and Traceable Requirement Engineering Process (STREP)" that will address all the three shortcomings just described.

4. Reasons to Propose the STREP Method

Dealing with security issues during the development of systems, is highly essential, because security of our systems is becoming increasingly important in our

society. Now a day, the security threats are increasing in frequency, spread, seriousness. The Internet is one major medium for different types of security threats. People are increasingly making security attacks to systems from outside organization, especially using the Internet. Nature and number of attacks is increasing day by day. Security threats may also occur from inside the organizations.

The major reasons to propose the STREP method is not just that to show that the security threats are increasing day by day. But rather the major reasons to propose STREP method is to emphasize the fact that, just like any other attribute of software, the security should also be addressed at starting stage of the development lifecycle [12]. The security should be considered as an essential attribute for the system to-be-developed and addressed immediately at the early stages.

The proposed STREP method not only deals with security issues of the system at requirement engineering stage, but also makes the security requirements more traceable to be used at later stages of development lifecycle, and as a result, secure systems are produced that are also usable as the customer wishes.

5. The STREP Method

The STREP (Secure and Traceable Requirement Engineering Process) collects the software requirements just like the conventional requirements engineering processes do *i.e.* eliciting requirements, analysing requirements, specifying requirements, but in a modified way such that the security requirements are also captured. The STREP also documents the security threats to the system and misuse scenarios at very early stages (*i.e.* requirement engineering stage) of the system development lifecycle. Moreover, STREP also helps in extracting testing components from the security requirements. The STREP method has been shown in **Figure 1**.

The conventional requirement engineering method has been modified to STREP by adding two additional stages; STREP security requirement stage and STREP security testing stage. At a higher abstraction level, the advantages and prominent features of STREP are;

1) It is helpful for experts that are not well known with security and also helpful for customer, in the requirement specification stage.

2) It helps in creation of security components while dealing and addressing with security issues at requirement specification stage.

3) It helps in extracting testing components from the security requirement specified earlier through it.

STREP method has been shown in **Figure 1** and described in the followings.

5.1. STREP Security Requirements

The security requirement phase of STREP plans and specifies security requirements by adopting a holistic approach at requirement engineering stage. The main components of STREP security requirement portion are as follows;

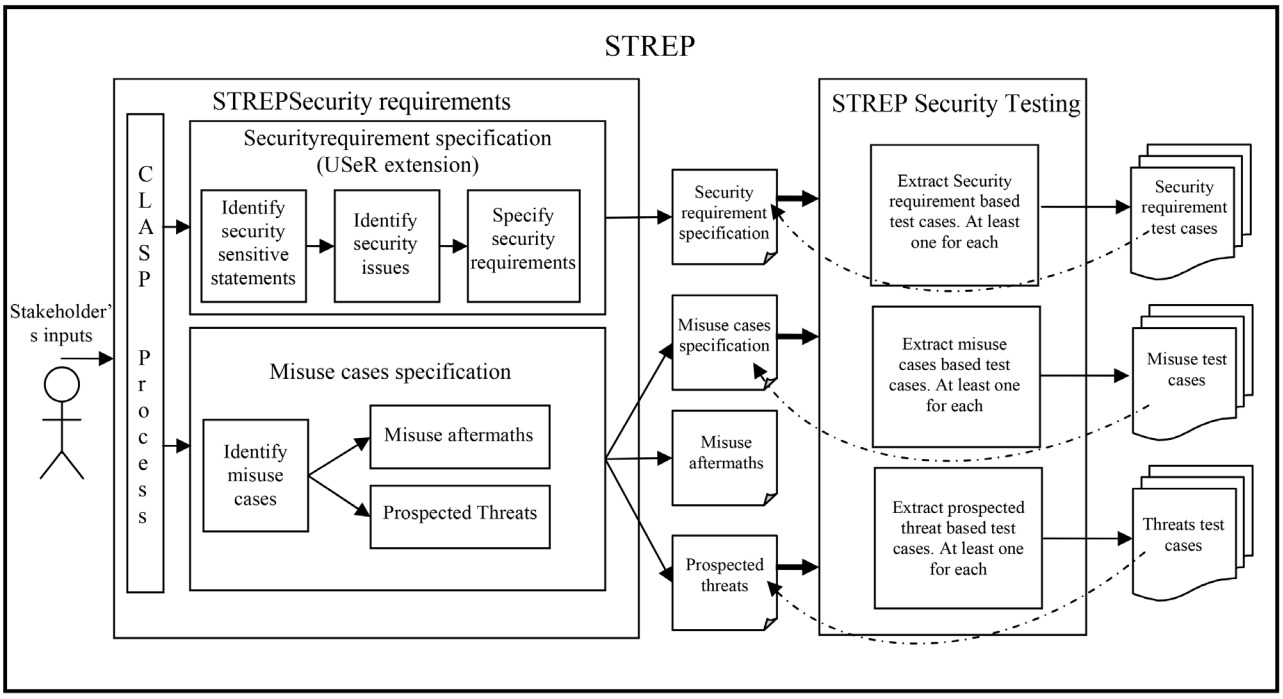

Figure 1. The STREP process.

1) *Specification Planning:* To specify the security requirements for different projects, STREP plans to uses the CLASP method discussed in [5]. CLASP defines 30 activities. In our STREP method, we plan to select some of them which are more important for security-sensitive projects. After the specification planning phase, the developers have set of tasks (CLASP activities) that will be executed in order to complete the requirement specification phase in a successful and secure manner.

2) *Security Requirements Specification:* The Security requirements are specified in three steps as shown in **Figure 1**. Each step refines the specification process. These three steps have been derived from USeR method [13]. The description of these three steps is as follows:

- *Security Sensitive Statements:* We find that the customers are not fully aware of security issues of system. Requirement engineers and customers normally specify requirement with respect to the functional aspect. At this stage, functional requirements are reviewed to identify and extract the security requirement.
- *Security Issues:* Security requirements identified in step 1 help the developers to know the issue associated with the functional requirements. One or more issues are detected from the security statements.
- *Security Requirement:* When the security issues are reviewed, they help in specifying the security requirements. Each security issue can help in specifying one or more security requirements [14].

By applying these three steps, the simple security statements are turned into security requirements. These three steps also help the developers, who are not

security experts themselves, to specify the security requirements.

3) *Misuse Cases Specification:* The STREP method also uses the *"Misuse Cases"* technique to provide assistance in security requirement specification. Misuse cases propose an inverse approach to specify the security requirements. The inverse approach is to think in term of attacks to harm the system to achieve malicious goals. Each identified misuse case gives two new security related information; aftermaths and threats.

In misuse aftermaths, we consider the consequences that a successful attack could have to the system. Considering the consequences assists the system developers in prioritizing the security requirements. It also helps to know the affects of a successful attack on the other modules of the system.

In misuse threats, we consider the possible threats to the system resources. Considering the security threat at the requirement engineering stage alarm us about different possible ways a system can be illegally accessed or harmed. This early consideration will help the developers make the affects of these threats less severe.

5.2. STREP Security Testing

As described earlier, one of the goals of STREP is to make the security requirement specification traceable and usable at later stages of the system development lifecycle. STREP decides to assist the developers at testing stage. In doing so, STREP guides the developers in extracting the test case from security requirement specification.

STREP introduces three types of test cases; security requirement based test cases, misuse case based test cases, and prospected threat based test cases.

1) *Security Requirement Based Test Cases:* These are the first type of test cases that are extracted from the requirement stage using STREP. Security requirements have information that, on a focused consideration, gives us one or more security test cases.

2) *Misuse Cases Based Test Cases:* Misuse cases give us information of how a requirement can be misused and illegally accessed using different ways. This information helps the developers to derive the test cases. The information in each misuse case can produce one or more test cases.

3) *Prospected Threat Based Test Cases:* STREP identifies different possible threats to requirements. These prospected threats are used to derive test cases. One is more test case are extracted from each prospected threat.

6. Contributions of the Proposed Method

Existing approaches for security requirement specification have variety of shortcomings [11]. The main shortcoming is that they do not provide sufficient support for security requirements to be traced and used at the later stages of development lifecycle. To our knowledge, STREP is the first method that not only specifies the security requirement but also helps in make them traceable and us-

able at the later stage of development lifecycle. Keeping the mentioned point in mind, some of the contributions of the STREP method are:

1) Introduction of a new approach for software requirement engineering, which increases the traceability and usability of security requirements.

2) Assistance for mapping the security requirements to design, implementation, and testing stages of development lifecycle.

3) Introduction of a kind of relationships between security requirements.

4) Introduction of a testing mechanism on the basis of specifications of requirements.

5) A platform for bringing the stakeholders closer.

A similar mechanism as in STREP but with a different focus can open a way to introduce new SRE methods for systems other than security-sensitive systems. For example, if we shift our focus from security to time, we can introduce a STREP like new SRE method for real-time systems.

7. Conclusions

Security is very important in software projects. Most of the current security requirement approaches do not seem adequate for applicability due to the reason that these approaches do not provide sufficient support for mapping the security requirements to the later stages of development. Secure and Traceable Requirement Engineering Process (STREP) is our proposed method which is not only helpful in security requirement specification at requirement engineering stage but also provides support for using the specified security requirements at later stages of development.

We have completely outlined the STREP method and our next task is to evaluate how the STREP is more effective as compared to other major SRE approaches. We also have a plan to perform a larger evaluation of the STREP method so that we may quantify its support for making security requirements specification more usable, useful, and understandable.

References

[1] Høegh, R.T. (2006) Usability Problems: Do Software Developers Already Know? Aalborg University Department of Computer Science, Aalborg East, DK-9220, Denmark.

[2] De Landtsheer, R. and van Lamsweerde, A. (2005) Reasoning About Confidentiality at Requirements Engineering Time. Département d'Ingénierie Informatique, Université catholique de Louvain B-1348 Louvain-la-Neuve (Belgium).

[3] Romero Mariona, J. and Richardson, D. (2009) Security Requirements Engineering: A Survey. University of California, Irvine.

[4] Whittle, J. and Wijesekera, D. (2008) Executable Misuse Cases for Modeling Security Concerns. Federal Railroad Administration 1120 Vermont Ave Washington, DC 20590.

[5] Viega, J. (2005) Building Security Requirements with CLASP. Secure Software, Inc. 2010 Corporate Ridge, Suite 820 McClean, VA.

[6] Futcher, L. and von Solms, R. (2008) Guidelines for Secure Software Development.

Nelson Mandela Metropolitan University P O Box 77000, Port Elizabeth, 6031 South Africa.

[7] Software Quality Attributes and Trade-Offs. (2005) Authors: Patrik Berander, Lars-Ola Damm, Jeanette Eriksson, Tony Gorschek, Kennet Henningsson, Per Jönsson, Simon Kågström, Drazen Milicic, Frans Mårtensson, Kari Rönkkö, Piotr Tomaszewski.

[8] Constantine, L.L. and Lockwood, L.A.D. (2003) Usage-Centred Software Engineering: An Agile Approach to Integrating Users, User Interfaces, and Usability into Software Engineering Practice. University of technology, Sydney (Australia), Constantine & Lockwood, Ltd.

[9] Beznosov, K. and Kruchten, P. (2004) Towards Agile Security Assurance. University of British Columbia 2356 Main Mall Vancouver, BC, V6T 4Z1 Canada.

[10] Luckey, M., Baumann, A. and Méndez, D. (2010) Reusing Security Requirements Using an Extended Quality Model. University of Paderborn, Paderborn.

[11] Mead, N.R. and Stehney, T. (2005) Security Quality Requirements Engineering (SQUARE) Methodology. Carnegie Mellon University 5000 Forbes Avenue Pittsburgh.

[12] Ardi, S., Byers, D. and Shahmehri, N. (2006) Towards a Structured Unified Process for Software Security. Department of Computer and Information Science Linköping University, SE-58183 Linköping, Sweden.

[13] Romero-Mariona, J. and Ziv, H. (2009) Later Stages Support for Security Requirements. University of California, Irvine Donald Bren School of Information and Computer Sciences.

[14] Faily, S. and Fléchais, I. (2010) A Meta-Model for Usable Secure Requirements Engineering. Oxford University Computing Laboratory Wolfson Building Oxford OX1 3QD, UK.