A MODEL BASED SECURITY REQUIREMENTS ENGINEERING FRAMEWORK

P. Salini
Research Scholar
Department of Computer Science and Engineering
Pondicherry Engineering College, E-mail: salini@pec.edu

S. Kanmani
Professor and Head
Department of Information Technology
Pondicherry Engineering College, E-mail: kanmani@pec.edu

ABSTRACT

Security engineering is a new research area in software engineering that covers the definition of processes, plans and designs for security. The researchers are working in this area and however there is a lack in security requirements treatment in this field. Requirements engineering is a major action that begins during the communication activity and continues into the modeling activity. Requirements engineering builds a bridge to design and construction. The security requirements is one of the non functional requirements which acts as constrains on the functions of the system, but our view is that security requirements to be considered as functional requirements and to be analyzed during the earlier phase of software development i.e. Requirements engineering phase. An increasing part of the communication and sharing of information in our society utilizes electronic media. Many organizations, especially distributed and Net-centric are entirely dependent on well functioning information systems. Thus IT security is becoming central to the ability to fulfill business goals, build trustworthy systems, and protect assets. In order to develop systems with adequate security features, it is essential to capture the corresponding security needs and requirements. It is called as the Security requirements engineering, which is emerging as a branch of software engineering,
spurred by the realization that security must be dealt with early during requirements phase. In this paper we have proposed a framework for Security Requirements Engineering and comparison is made with other Security Requirements Engineering methods.

**Keywords:** Security Engineering, Security Requirements Engineering, Security Requirements Engineering Framework.

**I. INTRODUCTION**

In recent years, reports of software security failures have become commonplace. Many proposals that incorporate security in the software engineering process. At one end of the spectrum, such proposals ensure good coding practices [2,5]. At the other extreme, the emphasis is on securing the organization within which a software system functions [1]. In either case, modeling and analysis of security requirements has become a key challenge for Software Engineering [3, 4]. When building secure systems, it is instrumental to take security concerns into account right from the beginning of the development process. It is well recognized in the software industry that requirements engineering is critical to the success of any major development project. The elaboration, specification, analysis and documentation of application-specific security requirements is an area that has been left virtually unexplored by requirements engineering research to date. Haley and his colleagues have given a framework for security requirements engineering, for elicitation and analysis. In the framework they claim that adequate security requirements must satisfy three criteria. The first criterion is definition: One must know what security requirements are. The second is assumptions: Security requirements must take into consideration an analyst’s implicit or explicit assumption that an object in the system will behave as expected. The third is satisfaction: One must be able to determine whether the security requirements satisfy the security goals and whether the system satisfies the security requirements. The system context is described using a problem-oriented notation, then is validated against the security requirements through construction of a satisfaction argument. The satisfaction argument consists of two parts: a formal argument that the system can meet its security requirements and a
structured informal argument supporting the assumptions expressed in the formal argument.

The Source of Security Problem are not considering Security requirements of complete system and not considering Security in the application itself. The Security Goals [6,7] such as confidentiality, integrity, accessibility, and accountability and combining management control principles and application business goals forms the Security requirements.

In this paper we present a framework for Security Requirements Engineering. The remainder of this paper is structured as follows: Section 2 reviews related work, background, motivation and challenges to Security Requirements Engineering; Section 3 introduces the framework and describes Security Requirements Engineering framework. Section 4 presents the comparison between our Security Requirements Engineering framework with Haley and his colleagues framework for security requirements methods, while Section 5 concludes with future works.

II. BACKGROUND AND RELATED WORKS

We explore in this section Security Requirements, its issues and Haley and his colleague’s framework for Security Requirements. We provide the related works and challenges to web applications.

A. SECURITY REQUIREMENTS

Security Requirements is defined as constraints on the functions of the system, and these constraints operationalize one or more security goals. But most requirements engineers are poorly trained to elicit, analyze, and specify security requirements, often confusing them with the architectural security mechanisms that are traditionally used to fulfill them. They thus end up specifying architecture and design constraints rather than true security requirements. This paper presents the different types of security requirements and provides associated examples with the intent of enabling requirements engineers to adequately specify security requirements without unnecessarily constraining the security and architecture teams from using the most appropriate security mechanisms for the job.
The engineering of the requirements for a business, system or software application, component, or (contact, data, or reuse) center involves far more than merely engineering its functional requirements. One must also engineer its quality, data, and interface requirements as well as its architectural, design, implementation, and testing constraints.

Whereas some requirements engineers might remember to elicit, analyze, specify, and manage such quality requirements as interoperability, operational availability, performance, portability, reliability, and usability, many are at a loss when it comes to security requirements. Most requirements engineers are not trained at all in security, and the few that have been trained have only been given an overview of security architectural mechanisms such as passwords and encryption rather than in actual security requirements. Thus, the most common problem with security requirements, when they are specified at all, is that they tend to be accidentally replaced with security-specific architectural constraints that may unnecessarily constrain the security team from using the most appropriate security mechanisms for meeting the true underlying security requirements. This paper will help to distinguish between security requirements and the mechanisms for achieving them, and will provide you with good examples of each type of security requirement. In today’s world of daily virus alerts, malicious crackers, and the threats of cyber terrorism, would remember the following objectives of security requirements.

**B. SECURITY REQUIREMENTS ISSUES**

In reviewing requirements documents, we typically find that security requirements, when they exist, are in a section by themselves and have been copied from a generic set of security requirements. They tend to be general mechanisms such as password protection, firewalls, virus detection tools, and the like. The requirements elicitation and analysis that is needed to get a better set of security requirements seldom takes place. Even when it does, the security requirements are often developed independently of the rest of the requirements engineering activity and hence are not integrated into the mainstream of the requirements activities. As a result, security
requirements that are specific to the system and that provide for protection of essential services and assets are often neglected.

Current research recognizes that security requirements are negative requirements. General security requirements, such as “the system shall not allow successful attacks,” are therefore generally not feasible, because there is no agreement on ways to validate them other than to apply formal methods to the entire system, including COTS components. We can, however, identify the essential services and assets that must be protected. We are able to validate that mechanisms such as access control, levels of security, backups, replication, and policy are implemented and enforced. We can also validate that the system will properly handle specific threats identified by a threat model and correctly respond to intrusion scenarios. [8]

C. HALEY AND HIS COLLEAGUES FRAMEWORK FOR SECURITY REQUIREMENTS

Haley and his colleagues have given a framework for security requirements engineering, for elicitation and analysis.

A framework for Security Requirements Engineering has the following 4 activities done in every iteration. [1]. Iteration between requirement and design activities is an important part of the framework.

Stage 1: Identify Functional Requirements

The only requirement the framework places upon this stage of the development process is that a representation of the system context be produced. How the requirements engineer gets to this point is left open.

Stage 2: Identify Security Goals

There are three general steps required to identify the security goals: identify candidate assets, select the management principles to apply, and then determine the security goals. The result is a set of security goals which are validated by ensuring that the business goals remain satisfied.

The first iteration through this step results in the generation of primary security goals. Subsequent iterations result in secondary security goals, which are traceable,
perhaps through multiple levels and through security requirements, to the original, primary, security goal(s).

**Stage 3: Identify Security Requirements**

Define security requirements as constraints on functional requirements that are needed to satisfy applicable security goals. To determine the constraints, we must determine which security goals apply to which functional requirements, which means we must know which assets are implicated in fulfilling a particular functional requirement. In the same fashion as security goals, the first iteration through this step results in primary security requirements. Subsequent iterations generate secondary security requirements.

**Stage 4: Construct satisfaction arguments**

It is important to verify that the security requirements are satisfied by the system as described by the context. To convince system can satisfy the security requirements, the outer argument, consists of a formal argument to prove that the instance of the system under examination satisfies its security requirements, with two important assumptions: that the context is correct and that the implementation will not introduce any conflicting behavior. The inner argument consists of structured informal arguments to support the assumptions about system composition and behavior made in the formal argument. Satisfaction arguments assist with identifying security-relevant system properties and determining how inconsistent and implausible assumptions about them affect the security of a system. The framework doesn’t give any modeling information and it is a complicated method to be followed by the software engineers.

In requirements and software engineering, researchers have investigated methods for analyzing security requirements by using Security Quality Requirements Engineering (SQUARE) is a process aimed specifically at security requirements engineering [9], Gustav Boström and his colleagues consider security requirements engineering in a different context, agile development with a focus on extreme programming (XP) practices [10], Clasp is a major initiative for securing software development life cycles [11], Steve Lipner and Michael Howard describe the Microsoft Trustworthy Computing Security Development Lifecycle [12, 13, and 14], Axelle Apvrille and Makan Pourzandi suggest four steps for the security requirements and analysis phase [15] Security Requirements Engineering Process [16] , Eduardo Fernandez
-use cases are helpful for determining the rights each actor needs and for considering possible attacks. [17]. Gunnar Peterson -use and misuse cases as a basis for security requirements, with additional nonfunctional requirements. [18]. Kenneth van Wyk and Gary McGraw suggest using abuse cases. [19] Core security requirements artifacts [20] takes an artifact view and starts with the artifacts that are needed to achieve better security requirements. Security patterns are useful in going from requirements to architectures and then designs [21, 22, and 23]. Tropos is a self-contained life-cycle approach [24]. It is very specific in terms of how to go about requirements specification. Formal specification approaches to security requirements, such as Software Cost Reduction (SCR) [25] have also been useful. The higher levels of the Common Criteria [26] provide similar results. The security requirements framework was published in [28] and further refined in [29] and Threat descriptions. [30].

III. SECURITY REQUIREMENTS ENGINEERING

Software systems become more and more critical in every domain of the human society. Transportation, telecommunications, entertainment, health care, military, and education; the list is almost endless. These systems are used not only by major corporations and governments but also across networks of organizations and by individual users. Such wide use has resulted in these systems containing a large amount of critical information and processes which inevitably need to remain secure. Therefore, although it is important to ensure that software systems are developed according to the user needs, it is equally important to ensure that these systems are secure.

However, the common approach towards the inclusion of security within a software system is to identify security requirements after the definition of a system. This typically means that security enforcement mechanisms have to be fitted into a pre-existing design, leading to serious design challenges that usually translate into the emergence of computer systems afflicted with security vulnerabilities. Recent research has argued that from the viewpoint of the traditional security paradigm, it should be possible to eliminate such problems through better integration of security and requirements engineering. Security should be considered from the early stages of the
development process and security requirements should be defined alongside with the system's requirements specification.

The Security Requirements Engineering is the process of eliciting, specifying, and analyzing the security requirements for system fundamental ideas like "what" of security requirements is, it is concerned with the prevention of harm in the real world and considering them as constraints upon functional requirements.

Many methods have been developed that facilitate this kind of requirements analysis and the development of security requirements. The internet has already created social and economic opportunities for people around the world. But even there are many Challenges to Web Applications Security like threats, some of the threats like cookies, phishing, pharming spyware, worms, Trojans and virus which cause to denial of service hacking into and defacing web sites and destroying files.

A. SECURITY REQUIREMENTS ENGINEERING FRAMEWORK

Our framework follows the spiral process model which is iterative and all phases of Requirements Engineering is covered in security requirements engineering framework.

1 INCEPTION

Inception is to establish the ground work before to start the elicitation of security requirements. Different steps are involved in the inception phase of security requirements engineering.

1.1 Identifying the objective of the software system

The software system objective must be identified from the customer requirements who needs the computer based system.

1.2 Identifying the Stakeholders

The identification of stakeholders plays an important role in security requirements engineering. The stakeholders include the developer, customers/end users, security experts, requirements engineering team and other interested people. Each stakeholder is responsible to find the assets and security goals. The security experts help in finding the security requirements and security mechanisms to obtain high level of security to the software system. The stakeholders will have multiple view points on the security requirements of the system. It may be conflicting security requirements and the
stakeholders will help to prioritize the assets and security requirements of the system. So care to be taken to prepare the list of stakeholders, to improve the effectiveness of preliminary communication and collaboration between the stakeholders.

1.3 Identifying the Assets

The next step is to identify the assets of the targeted system. Assets may be business or system assets (e.g.: data, money, and password). From our survey it is found that assets identification is an important step in security requirements engineering. The assets should be identified in the context of the software system, so the objective of software system is to be identified first. To identify the assets different techniques like interview, questionnaire, and brainstorming can be used. The stakeholders’ helps in finding the assets. Assets should be viewed not only at developer or customer/end user perspective but also in attacker’s point of view. Assets can be identified from existing documents. The identified assets have to be categorized and prioritized with regard to different stakeholders need. Assets can be categorized under Confidentiality, Integrity and Availability and prioritized as low medium and high level of preference. Example password can be categorized under confidentiality.

Inception phase of security requirements engineering should be worked with high level of collaboration and care.

2 ELICITATION

The next phase in security requirements engineering is elicitation, the stakeholders and requirements engineering team work together identify the problem, propose the solution and specify the set of security requirements. There are different steps involved in the elicitation phase of security requirements engineering.

2.1 Select an elicitation technique

Before the elicitation phase starts some ground work to be done for selecting the elicitation technique. Some of the elicitation techniques are, misuse cases, Soft Systems Methodology (SSM), Quality Function Deployment (QFD), Controlled Requirements Expression (CORE), Issue Based Information Systems (IBIS), Joint Application Development (JAD), Feature-Oriented Domain Analysis (FODA), Critical Discourse Analysis (CDA), and the Accelerated Requirements Method (ARM). A suitable method
can be chosen from these elicitation techniques based on expertise choice, level of the security to achieved, cost–effort benefit and organizational policies.

2.2 Elicitation of non-security requirements and goals

The business goals are identified, and then the non-security requirements and goals of the software system are elicited. The collaborative requirement gathering is adopted to gather non-security requirements and goals. The non-security requirements are categorized as essential and non-essential requirements and prioritized according to the Stakeholders preference.

2.3 Generate use cases for the software system

The non-security requirements are gathered; for better understanding the use case of the software system should be generated. The use case is the set of scenarios that encompass the non-security requirements of the system created by the developers and users of the system.

2.4 Identify the security goals / security objectives

The security goals / security objectives can be identified with respect of assets, business goals and organizational principles that is the security policies of the organization. The list of security goals can be identified and the security goals can be main goals and sub goals. The main goals are the top goals, e.g. Confidentiality, Integrity and Availability, Anonymity that to be identified for the software system and the sub goals, e.g. Prevent tampering, trail and access control which comes under the top security goal Integrity. The techniques like Facilitated Application Specification Technique (FAST), survey and interviews can be used to identify the security goals / security objectives.

2.5 Identify threats and vulnerabilities

By identifying the assets, business goals and security goals the threads to the software system can be identified. The threats and vulnerabilities can be identified for each security sub goals, e.g. tamper threat, password attack, wire tap.

2.6 Risk Assessment

The next step is to assess and determines the risk when the threads and vulnerabilities occur. The impact of threads and vulnerabilities are analyzed and risk determination process is carried out. To do risk determination process any of risk
assessment tests model like National Institute of Standards and Technology (NIST) model [15], NSA’s INFOSEC Assessment Methodology [16], Butler’s Security Attribute Evaluation method (SAEM) [17], CMU’s “V-RATE” method [18], Yacov Haimes’s RFRM model [19] can be used.

2.7 Categorize and Prioritize the Threads and Vulnerabilities for mitigation

The threads and vulnerabilities can be Categorize with respect to the security goals and security policies of the organization and prioritized based on the level of security and assets to be secured, e.g. tamper threat Categorize under top security goals Integrity, unauthorized users under Confidentiality, and Integrity. This process can be done with help of the survey or interview between the stakeholders.

2.8 Generate misuse cases for the software system

The detailed set of misuse case of the software system should be generated that encompass the most significant threads to the system e.g. tamper misuse case, unauthorized users misuse case.

2.9 Identify the security requirements

The security requirements are the counter measures that the software system should have, as the functional requirements, e.g. Threat – password attack, wire tap, tamper, and Security goals – Availability - password attack, wire tap and Integrity - tamper. The Security requirements to prevent these threats are Prevent password attack, encrypt communication, authenticate, validate data and lock data.

2.10 Generate use cases for the software system considering security requirements.

The security requirements are gathered; for better understanding the use case of the software system should be generated, that encompass the security requirements of the system created by the developers and users of the system.

3 ELABORATION

3.1 Generate structural analysis models

In this phase of security requirements engineering different analysis models are developed. These models form the solid foundation for the design of security requirements. The data model, flow models and behavioral models are the structural analysis models.
3.2 Develop UML diagrams

Develop UML diagrams for detailed view of security requirements and for better understanding of the security based computer system. These diagrams can be used to generate coding and test cases for testing the security requirements.

4 NEGOTIATION AND VALIDATION

In this phase the security requirements are categorized as essential and non essential requirements and prioritized according to the level of security and Stakeholders preference of security requirements. Then rough effort time and cost are estimated to implement security requirements.

The validation is done by the security experts and engineers with the requirements of the stakeholders.

5. SPECIFICATION

This is the last phase in security requirements engineering. The security requirements specifications are made and they are validated with the stakeholders and this specification forms the source for the design of security requirements.

IV. COMPARISON OF THE FRAMEWORK

We applied Haley and his colleagues’ framework in the web applications which consider information as treasure or assets and evaluated the strength of the framework. During this first iteration, we establish the context for the system, the functional requirements, and the primary security goals and requirements.

Haley and his colleagues suggest artifacts that are probably too complex for regular developers [27,31]. In this framework there are many limitations like they don’t categorize or prioritize the security requirements, lack of coding standards and process planning. This may lead to complexity of adopting this framework. Framework follows iterative process which may add advantage to find new security requirements but this will lead to more complexity to the web applications. Our security requirements framework overcomes all the limitations of Haley and his colleagues’ framework.
V. CONCLUSION

In this paper we discussed about, Security Requirements and, Security Requirements Engineering. All the approaches of Security Requirements Engineering and its limitations were identified. In short, there is no single right answer when it comes to security requirements engineering. A lot depends on the processes that are already in place in a particular organization. Some organizations may prefer a detailed, specific method, whereas other organizations may prefer an approach that allows them to select methods to incorporate into existing processes. Another factor is the extent to which the project or organization is mission critical. This can dictate the level of formality used in requirements engineering and the need for assurance levels of security. We have developed a framework for Security Requirements Engineering covering the limitations of Haley and his colleagues’ framework and we have compared our framework with Haley and his colleagues’ framework. In future we have planned to apply our security requirements engineering framework for banks and web applications. The other future work is to develop a tool to elicit security requirements. We have also planned to develop a tool to generate test cases from the security requirements.

REFERENCES


