A Modified & Extended Security Quality for Requirements Engineering (SQUARE) Methodology into Standard Life-Cycle models

R.S.Sindhu Theja  
Sreenidhi Inst. of Science & Tech.  
Dept. of CSE, Hyderabad

G.Kalyani  
Sreenidhi Inst. of Science & Tech.  
Dept. of CSE, Hyderabad

E.Pragnavi  
Sreenidhi Inst. of Science & Tech.  
Dept. of CSE, Hyderabad

ABSTRACT
SQUARE (Security Quality for Requirements Engineering) is method for eliciting, categorizing and prioritizing security requirements in software development projects. SQUARE [1] provides security for the lifecycle models in the requirements phase that generates a final deliverable of categorized and prioritized security requirements. This paper analyzes the performance of SQUARE methodology with various applications (web applications) which provides more security. A detailed evaluation of all the nine steps along with subsequent reassessment and refinement is carried. The paper also examines the four major steps of SQUARE namely Risk assessment, Elicitation techniques [3], Categorization and Prioritization are reviewed with major characteristics. The above four steps are analyzed against application development. And finally modifications and the observations that are extended into complex application development are discussed.

Keywords:
Requirements Engineering, SQUARE methodology.

1. INTRODUCTION
Almost all of the software systems installed today involve security in one way or another. Yet these systems are breached. One of the most critical phases of Security architecture lifecycle is the security requirements engineering phase, because any flaws occurring in these phases will cost a lot to fix in the later stages. A possible solution lies in creating security architecture, ensures that security as a subsystem and creates separation of concerns. Without requirements security architecture cannot be built. SQUARE is one of the most popular Security requirements methodologies. It is a process aimed specifically at security requirements engineering. SQUARE Methodology is able to accurately adapt to the current state of the practice of software and requirements engineering. This paper mainly focuses on performance of SQUARE methodology and the four major steps risk assessment, elicitation, categorization and prioritization were analyzed against application development. Finally the modifications and observations are extended into complex application development.

2. SQUARE METHODOLOGY
SQUARE can be decomposed into nine discrete steps. Each step identifies the necessary inputs, major participants, suggested techniques, and final output. The following are the nine steps of SQUARE methodology.

2.1 Agree on Definitions
In order to guarantee effective and clear communication throughout the requirements engineering process, the requirements engineering team and stakeholders must first agree on a common set of terminology and definitions.

2.2 Identify Security goals
The purpose of Step 2 in SQUARE is for the stakeholders to formally agree on a set of prioritized security goals for the project. Without overall security goals for the project, it is impossible to identify the priority and relevance of any security requirements that are generated. In addition, the establishment of security goals scopes the rest of the SQUARE process. Initially, different stakeholders will likely have different security goals. The security goals of the stakeholders may also conflict with one another. The security goals of the project must be in clear support of the project’s overall business goal, which also must be identified and enumerated in this step.

2.3 Develop Artifacts
Before the requirements engineering team and stakeholders can generate a comprehensive set of security requirements, the team must collect a complete set of artifacts of the system. Example: System architecture diagrams, use case scenarios/diagrams, misuse case scenarios/diagrams, attack trees, standardized templates and forms.

2.4 Perform Risk Assessment:
The purpose of this step in the SQUARE process is to identify the vulnerabilities and threats that face the system, the likelihood that the threats will materialize as real attacks, and any potential consequences of an attack. Without a risk assessment, organizations can
be tempted to implement security requirements or countermeasures without a logical rationale. The risk assessment also serves to prioritize the security requirements at a later stage in the process.

2.5 Elicitation Technique
The requirements engineering team must select an elicitation technique that is suitable for the client organization and project. Although this task may appear to be straightforward, it is often the case that multiple techniques will likely work for the same project.

2.6 Elicit Security Requirements
This step is the heart of the SQUARE process: the elicitation of security requirements. To the benefit of the requirements engineering team, most elicitation techniques provide detailed guidance on how to perform the elicitation, so this step is simply a matter of executing the technique.

2.7 Categorize Requirements
The purpose of this step is to allow the requirements engineer and stakeholders to classify the requirements as essential, non-essential, system level, software level or as architectural constraints.

2.8 Prioritize Requirements
In most cases, the client organization will be unable to implement all of the security requirements due to lack of time, resources, or developing changes in the goals of the project. Thus, the purpose of this step in the SQUARE process is to prioritize the security requirements so that the stakeholders can choose which requirements to implement and in what order, software level, or as architectural constraints.

2.9 Requirements Inspection
The last step of the SQUARE process, requirements inspection, is one of the most important elements in creating a set of accurate and verifiable security requirements. Inspection can be done at varying levels of formality, from Fagan Inspections to peer reviews.

3. ANALYSIS OF MAJOR STEPS OF SQUARE METHODOLOGY

3.1 Risk assessment
It is the determination of quantitative or qualitative value of risk related to a concrete situation and a recognized threat (also called hazard). The following are some of the risk assessment techniques: National Institute of Standards and Technology (NIST) model, Yacov Haimes’s RFRM model, Martin Feather’s DDP model. Based on the comparisons made between these techniques NIST is found to be best model for making risk assessment.

3.2 Requirements Elicitation
The process of discovering the requirements for a system by communication with customers, system users and others who have a stake in the system development. The techniques mainly used are IBIS (Issue Based Information Systems), JAD (Joint application development), ARM (Accelerated Requirements Methodology). IBIS ranked as best elicitation technique based on the comparisons.

3.3 Categorization
In this process the requirements engineer and stakeholders classify the requirements as essential, non-essential, system level, software level, or as architectural constraints. A 3 dimensional analysis of requirements is performed by adding counter measure. It is a safeguard that addresses a threat and mitigates risk. So that for each type of requirement a counter measure is encountered. A counter measure is a safeguard that addresses a threat and mitigates risk so that for each type of requirement a counter measure is encountered. For example, the spoofing attack of user identity the different countermeasures encountered is strong authentication and session control.

Figure: 3.1 Dimensional representation of requirements
3.4 Prioritization

Once a set of security requirements are identified, usually prioritization should be done. Due to time and budget constraints, it can be difficult to implement all requirements that have been elicited for a system. Also, security requirements are often implemented in stages, and prioritization can help to determine which ones should be implemented first. A number of prioritization methods have been found to be useful in traditional requirements engineering and could potentially be used for security requirements. Some of the techniques are AHP (Analytical hierarchy process), Triage, Theory-W, Binary Search Tree. BST is found to be best prioritization technique as it can prioritize even if there are more requirements.

3. ANALYSIS AGAINST A WEB APPLICATION

Security requirements should be reflected in both business and functional requirements. At the business requirement level, one generally identifies what resources need to be protected. Focuses on demands from the customer and demands that are internal to the organization. Functional requirements should specify what mechanisms should be put in place to provide security services on resources. The Web application model, like many software development models, is constructed upon 3 tiers: User Services, Business Services and Data Services. Generally, requirements should specify necessary authentication factors and methods for each endpoint on a communication channel.

In this paper, the four major steps of SQUARE are analyzed against a web application. The following are the different ways of analysis.

- Security risk assessment have become mission critical tasks i.e., when it comes consideration with group of representatives.
- How do organizations perform an accurate security risk assessment of their IT systems and the critical information they store? Risk surrounds us every day but it’s not so easy to comprehend Web security risk management: How much does it actually cost a company when a Web server is breached, or if an attack disrupts the availability of critical Web systems? What are the costs associated with a hacker or competitor snatching proprietary information or customer lists from an insecure Web server? How Web security risk management is performed depends entirely on knowing the answers to these questions. Such risks can be seen more clearly through the following simple equation that quantifies a security risk assessment:

\[
\text{Risk} = \text{Value of the Asset} \times \text{Severity of the Vulnerability} \times \text{Likelihood of an Attack.}
\]

All of these major problems usually are the result of a lack of due care within the Web application development and maintenance processes. In organizations where security is not ‘baked in’ to both the business planning and application development processes, there can be an appalling lack of awareness of the need to incorporate security best practices from day one.

The best way to avoid such disasters is to establish an ongoing security risk management process that begins with quantifying the value of Web applications, as well as the data they manage, through a complete security risk assessment. There are several risk assessment techniques to get rid of different types of risks. Organizations then must continuously identify and mitigate the vulnerabilities and risks associated with those systems from the beginning and throughout their lifecycle: from development through production.

In this report NIST model is used for risk assessment. Here the output concentrates more on attacker’s motives once enter inside the system. This model consists of: Threat Identification, Vulnerability Identification, Control Analysis, Likelihood Determination, Impact Analysis, and Risk Determination.

The next major step is Elicitation technique. This paper reports a systematic review of empirical studies concerning the effectiveness of elicitation techniques, and the subsequent aggregation of empirical evidence gathered from those studies. The most significant results of the aggregation process are as follows: (1) Interviews, preferentially structured, appear to be one of the most effective elicitation techniques; (2) Many techniques often cited in the literature, like card sorting, ranking or thinking aloud, tend to be less effective than interviews; (3) Analyst experience does not appear to be a relevant factor; and (4) The studies conducted have not found the use of intermediate representations during elicitation to have significant positive effects.

Categorization is the process which allows the requirements engineer and stakeholders [3] to classify the requirements as essential, non-essential, system level, software level, or as architectural constraints. Here in this paper another type of dimension is added i.e., counter measure which is an action, process, device or system that can prevent or mitigate the effects of threats to a computer. Different types of counter measures are authentication, authorization, session management, input validation, logging, cryptography, error handling.

Prioritization technique is used in order to prioritize requirements. Among all the prioritization techniques BST is found to be best. Because of its easy understanding, takes less time and even if there are more requirements the prioritizations is easy. Based on these
concerns BST (Binary Search Tree) is used in order to prioritize requirements. It is rated more accurate, better to scale up and easiest method to use.

4. MODIFICATIONS AND OBSERVATIONS

Many organizations, including organizations that have already invested considerable resources to safeguard system security are unprotected against a new breed of security attacks. Security measures such as implementing firewalls, authentication and access control systems, network intrusion detection/prevention systems, anti-virus solutions, anti-spyware solutions, and Secure Sockets Layer are indeed critical. Yet, these safeguards alone are no longer sufficient. Attackers now expect these basic security safeguards to be in place, and attack the system in a new way: via internally-developed custom applications.

Even if an application is protected against standard application security attacks (including SQL injection, parameter manipulation, buffer overflows, cross-site scripting, and so on) it may still be vulnerable to a dangerous new breed of specialized application attacks: attacks that take advantage of flawed application logic. The most common approach to securing applications is not the optimal way to prevent standard application attacks, and it does little to prevent the application logic attacks that are becoming attackers’ tool of choice. The industry's common response to securing applications has been to try to test security into the application at the end of the development process. But this approach fails to address the root cause of the problem: security, like quality, must be built into the application.

Building security into an application involves designing and implementing the application in a way that reduces the risk of security attacks, then verifying that the policy is implemented and operating correctly. Essentially, security becomes a specification issue. If the specification does not define how the application should be built to safeguard security, the application will be vulnerable to the types of sophisticated attacks that are starting to emerge now and will become increasingly prevalent in the future, when common application vulnerabilities are protected and the application logic becomes attackers’ only real entryway into the system. And if that specification is not a living document that is actually implemented in the code, the organization will be as vulnerable to attacks as it would without a security policy.

The security policy consists of core rules based on application security best practices. Application-specific rules that address the organization's specific security concerns are added to extend and customize the core set of rules. This process produces a document that enforces the centralization of security mechanisms, prevents coding problems that can lead to security vulnerabilities, and details custom security requirements. Security policies are adopted by security experts, such as OWASP [2], and are mandated for compliance with many regulations. In this project, web application is provided security by using SQUARE methodology.

5. CONCLUSION

Security requirements are of paramount importance, as they directly reflect the quality of the product. Current processes support capturing requirements like security, but doing so is often an afterthought relative to functional (end user) requirements. Detailed scenarios for quality attributes are not captured during the requirements phase. For instance, financial projects have critical security issues that need to be elaborated in security requirements. The four major steps of SQUARE i.e., risk assessment, elicitation, categorization and prioritization are analyzed and found best techniques in them by making comparisons of the techniques with each other. This report can help any organization to use SQUARE to elicit security requirements in security-critical projects.

In Risk assessment NIST model is analyzed as best technique of finding risks. In the process of elicitation by considering different criteria all the techniques are compared and found IBIS as best elicitation technique for discovering new requirements from requirements engineering, customers and stakeholders. The original SQUARE methodology consists of two dimensional representation of requirements categorization. This work suggests a new dimension called ‘countermeasure’, to form a three dimensional representation of security requirements. In the process of prioritization Binary search tree is found to be best technique in order to prioritize requirements. These are modifications and extensions made for SQUARE methodology.

6. FUTURE WORK

This work is extended to implement and develop a financial web application in J2EE environment and security is a major concern for all the requirements. The application requirements will be analyzed with modified SQUARE methodology.

7. REFERENCES


