CHAPTER I

INTRODUCTION

1.1 Overview

Everyone agrees that security is a big problem nowadays as common attacks against computer software are increased day to day. The security attacks in the modern economies can cause financial damages. They can even threaten human lives, due to the increasing interconnection of systems, some attacks can be anonymous and from remote locations. Thus, computer's software needs to be developed with the consideration of security requirements from the early phases of Software Development Life Cycle (SDLC), especially in the requirement phase of the software building. Futcher (a), 2011 state that when you add an important factor after completing system, it will not be as effective as if it was from the beginning of the process at the system step by step (Futcher (a), 2011).

In the SDLC process, it is very important for the designer to understand the vulnerabilities and potential errors presented in every phase. Especially in the requirement phase, due to its importance in the SDLC process. It specifies the core of the proposed system. As well as it helps to understand the amount of the destruction that the weaknesses can cause to different features of the system. Furthermore, experts have also to realize and quantify the security requirements that should be incorporated in order to address these vulnerabilities. The information measured according to the perspectives of vulnerabilities is significant. If the problem leads to vulnerability is not fixed in an early phase, the cost of solving might raise tenfold with each and every additional at the development phase (Khan, 2008).

Thus, this research aims to reinforce: (a) adapting the method of appreciative inquiry in the evaluation of the Security Requirements (SR), and (b) using the mathematical model to quantify Security Requirements Index (SRI). The idea of the research is to establish the security requirements that allows better development of the software in terms of security and does not waste unnecessary resources of the stakeholders.

1.2 Background

Most of the time, security is not addressed from the very beginning of a software development, and it is only incorporated after the software has been developed, which eventually leads to vulnerabilities in the software product (Khan, 2008; M. Khan & Zulkernine, 2008; Hans, 2010; Futcher, 2011). Software often is developed with minimal concern for security according to Viega and Mcgraw, (2001). In addition, programmers also usually do not take into account from the beginning of the SDLC about the security issue (Howard & LeBlanc, 2003).

Adding appropriate security later is difficult, if it is even impossible, because the analysis or design itself may be fundamentally insecure. Moreover, it can also cause changes to the features and the application that affect backwards compatibility (R. Khan, 2011). In examining some real-world experiences, project managers faced project "fires" when risks were not identified during the beginning phases. Security is a necessary property from the beginning of the software life cycle (Hans, 2010). It is estimated that 70 percent of reported security incidents result from exploits against defects in the analysis and design or code of software. (Wang & Zhang, 2008; Wang et al., 2009).

Security is still considered as something that 'gets in the way', while given little importance to the security aspects of the project or financial support to this aspect, they expect that this can save money and ignore the things redundant (Banerjee, 2009; Futcher (a), 2011). It is impossible to build a 100 percent secure system, especially in system development where a trade-off is made between security and other constraints such as the cost and time (Gregoire et al., 2007; Hans, 2010). While risk cannot be completely eliminated, a secure application can identify and mitigate these risks to a

more manageable level (Hanny, 2010) by improving the software, and reduce the software vulnerabilities. Also, it has been observed that if security is implemented right from the inception of software, it saves the economy billions of dollars (Hans, 2010). According to Banerjee (2009) and Hans (2010) both research and real-world experience indicate that correcting weaknesses and vulnerabilities as early as possible in the software's life cycle is far more cost-effective over the lifetime at the software than the development which makes the software not vulnerable and fault free as possible. In most cases security requirements should be the foundations and rules of the development, being essential complement stages of the SDLC from the start until the end of the process.

Security compromises continue to rise. Hackers found a new attack vector and are successfully exploited it to breakdown the targeted software. Therefore, it is truly said by Gene Spafford (2010), "Security is like adding brakes to cars. The purpose of brakes is not to stop you: it's to enable you to go fast!".

Secure software does not happen by accident. It is accomplished only when every analyst, designer, developer, tester and manager working on a project take security seriously and at each phase of SDLC (Banerjee, 2009). Security should be everywhere as it begins at the requirements phase and should be on the mind of every person during the entire development processes, with developer's knowledge of each phase and what security aspects should be contain; quantified information about vulnerabilities and risks that can be possibly faced in future (Banerjee, 2009). Therefore, the developer should be able to know about source of software vulnerabilities and errors, so that these can be removed as soon as possible.

1.3 Problem Statement

According to Merlin (1999), the problem of developing software is not a new challenge, it exists since 1960s. Hence, the development problems eventually delayed a lot of projects. Moreover, these problems have increased the budgets, and despite this over expenditure, most of the software failed to meet the expectations of clients. Nevertheless, if the requirements engineers do not possess adequate knowledge about

the functionalities and the features of various elicitation methods, they will not be able to choose the appropriate elicitation methods for gathering user requirements, which might end up in development disasters (Gonzales, 2011b). The erroneous requirement documents will set the inappropriate goals in the product development phase. Hence, the erroneous requirement documents fail to fulfill the expectations of the customers and the proposed services (Gonzales, 2011b). Furthermore, amending the requirements in the mid-project development phase might end up in impediment and augmented cost.

From the literature studies, it is evident that a consistent approach to provide secure software is needed. Viega, (2004), Paul, (2008) and Futcher, (2011a) claimed that software developers generally focused on core functionality and features, where the security was typically only addressed as an afterthought and too late. Thus, software and system developers require practical and systematic approaches to obtain sufficient and credible evidence of the security level in the system under development (in early phases) in order to guide their efforts and ensure the efficient use of resources (Savola et al., 2012).

As a date, there is no reliable concrete technique or method to quantify security requirements in software industry (SDLC artifact) (Khan, 2008; Khan, 2011; Savola et.al. 2012). The quantified information about vulnerabilities is important because if an error leads to vulnerability was not corrected in an early phase, the cost of correcting it might increase tenfold with every additional developmental phase (Khan, 2008).

Furthermore, it is essential for the developer to know, in early phases of developing any software, how many vulnerabilities are presented, what is the potential damage vulnerabilities can cause to various assets of the system where the software is going to be a part of and what security requirements have to be incorporated to remove these vulnerabilities. However, very little work has been done to help the developer in this case (Khan, 2008; Khan, 2011). This will have an impact on the accuracy with which certain activities are being executed and on the set of the activities that will be executed in the first place.

According to Savola et al.(2012) The challenges include the extra work needed for metrics development and the lack of advanced tools to support security in requirements phase. From here, the arising problems faced by the developer can be checked and corrected as follows:

a. A successful secure system will be dealt with security aspects in early phase (requirements phase). It will enable the organization to identify and remediate risks with applications If implanted and executed effectively (Hanny, 2010), by using appropriate method to elicit and quantify security requirements and start the process of eliciting and quantifying security requirements with starting of elicit functional requirements.
b. The vulnerability assessment measurements offer early visibility of security effectiveness and efficiency during security critical phase of requirements. Early visibility helps especially in reducing gaps and biases in security correctness with respect to security effectiveness in future phases (Savola et al., 2012).

Thus, effort should be dedicated to elicit and quantify security requirements that will be positive effects on the future software; it will be more secure, especially. Using specialized approach in elicit security requirements like Appreciative Inquiry approach (AI), that will facilitate in improving the eliciting software requirements stage (Gonzales & Leroy, 2011). Therefore, this study aims to elicit and quantify security requirements in any SDLC, and emphasizes security as a functional requirements specification.

1.4 Research Questions

a. What is the best approach to elicit security requirements?
b. How to elicit security requirements at the early phase of SDLC?
c. How to quantify security requirements?
d. How the proposed framework helps in eliciting and quantifying security requirements?

1.5 Research Objective

The main objective of this study is to propose a secure software development methodology to elicit and quantify security requirements. The specific objectives are:

a. To construct a framework to elicit software security requirements.
b. To formulate a Quantification Technique for measuring software security requirements.
c. To evaluate the proposed framework using real case study with penetration testing and validation by security experts on a built prototype.

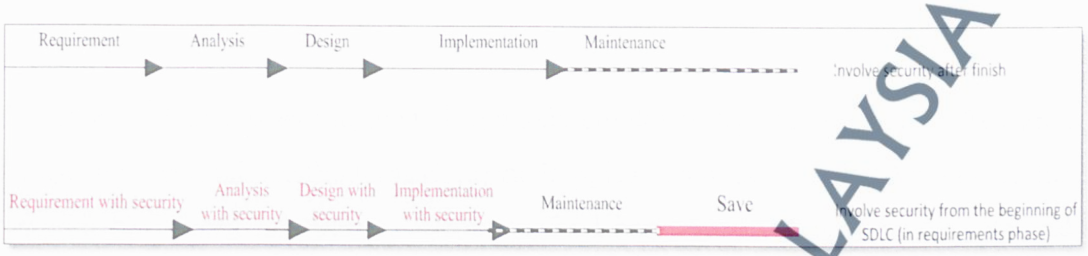
1.6 Significance of Study

As a matter of fact, there are a wide range of techniques that have been employed in requirements elicitation phase (Goguen & Linde, 1993). Each technique comprises quite a number of strengths and contributes towards the understanding of the designer pertinent to the organizational domain in a wide range of ways. A set of techniques have been defined and tested by requirements engineering experts, to make the requirements elicitation step more efficient and precise (Escalona & Koch, 2004). However, each requirements elicitation technique comprises certain limitations.

The importance of security requirements cannot be understated and accurate security requirements are deemed to be a necessity. This study is aimed at understanding the security requirements and how to successfully gathered these security requirements in software development. Thus, the software over networks has sensitive information and money transformation is applied depending on these software. It may also demonstrate how software stakeholders may obtain adequate and dependable indications of security effectiveness as early as possible in product development, because the ability to understand the software on behalf of the stakeholders is very important it and enable the developers to evaluate the security in the software, to mitigate effort, time and cost, by involving security aspects from early phase (requirements phase) of

software industry using security requirements elicitation technique. Figure 1 shows a comparison between the conventional SDLC and the proposed approach in this study.

FIGURE 1: Software Timeline.



1.7 Scope and Limitation of the Study

This study, in general will limit itself to a few real cases in terms of the proposed integrated method and conduct heuristic evaluation on the proposed integrated security requirements elicitation technique. Focus of this study will be on how to improve the quantification of security requirements in requirements phase integrated with Appreciative Inquiry approach and fuzzy soft set theory. The outcome is to develop a security requirement elicitation and quantification framework. Where the main interest is in the requirements elicitation phase.

According to the ignored steps in the SQUARE method and CLASP best practices the processes are not conducted due to the main objective in the proposed method that is elicit security requirements. The "Select Elicitation Technique" step has already been included in the proposed method that is considered as the main aim of this study. Some other ignored steps cover advanced stages of SDLC since the main focus is on the requirement phase.

1.8 Organization of the Thesis

The rest of this thesis is organized as follows: Chapter 2 discusses the necessary background and literature review about requirements engineering, security requirements and fuzzy theory; its structure, and its components. It shows the

relationship between normal/user requirements and security requirements and presents a comparison between methods and techniques used to elicit and quantify security requirements. The research benefits from this comparison. Also, AI is defined in this chapter.

The chapter 3 discusses the related methods employed by the researcher. Research design and research methodology, methodological choices and the process of the research have been introduced. A mix method has been chosen to investigate and develop a conceptual framework, which aims to identify the used techniques in eliciting and quantifying security requirements.

Chapter 4 develops the technique to elicit security requirements in the requirements phase which is named Secure Appreciative Inquiry Technique (SAIT).

Whereas chapter 5 contains the embedding of the fuzzy soft set algorithm with SAIT to gain Secure Appreciative Inquiry Fuzzy Quantification Technique (SAIFQT). SAIFQT is a technique to elicit each of the software and security requirements and quantify security requirements which are related to the proposed software through the SDLC artifact. Moreover, it is used to make the decisions for all elicited security requirements to prioritize and categorize them in the design phase of SAIFQT.

Chapter 6 provides a case study using SAIFQT. SAIFQT is used in the case study to elicit each of the software and security requirements and quantify security requirements, which are related to the proposed software (Price Website). Moreover, it is used to make the decision for all elicited security requirements that prioritize and categorize them in design phase at SAIFQT.

In chapter 7 findings and results that are shown in this part of the thesis. Also, the decision of the developers to deliver multi levels of security for any system that will be established. The result reveals the ability of the proposed technique in eliciting and

quantifying security requirements. Finally, providing research limitations and recommendations for future studies.

1.9 Summary

This chapter discusses the software security and SDLC process. Related preliminary works or background of security issues in requirement phase and methods employed by this study. A problem statement has been discussed, which is provide the evidences that are needed to create an approach or a method to provide secure software, and then research questions that will be solved through this research. Research objective has been established to find a solution for research problem, and to demonstrate the significance and scope of the study that is involved in the research topic.