CHAPTER-1
INTRODUCTION

The software development has transformed substantially from last two decades. Computer systems were usually islands in and of themselves, with limited or no connectivity. Applications were normally self-sufficient units and were installed after being verified for functionality on systems which were unconnected. At that time, software as a service did not exist except in the concept stage. Security was not deemed as important since violate primarily meant that the worst that could happen was the attackers could attack only themselves in a controlled environment. However, with the growth of the internet the paradigm shift in the way computer systems were networked and how the applications were developed, resulting in a momentous effect on security. The high degree of computer connectivity existing in today's society means that applications not designed to operate securely may susceptible to attack from both insiders and outsiders. Present day applications must be protected and must be anticipated to function in hypothetically antagonistic environment also (Mano Paul, 2008).

The biggest hurdle with today's software industry is to deliver a “Secure Software”. Security is of utmost vitality for the present day users. The modern day users demand best performance and accurate outputs in the least time possible from the application software. Besides this, they want the software to keep the data secure from being hacked. Thus, it is essential to design software that tends to perform security action for avoiding data loss or its theft or misuse in any respect. Software engineers and project managers remain puzzled to improve the aspects of security on the already developed software. To append security features in previously developed software will be a tedious and uneconomical task for any software developing unit. The important causes for this consist of: a) to include security attributes in application software, already in existence, is an intricate task. b) It is also not advisable to put at stake capital for the previously developed software. c) For software developers, it is not viable to deal with the roles of security experts (K. Beznosov and B. Chess, 2008).

Secure software is defined as the one which can resist any attack onto it. And to do so, it's a pre-requisite for all the application software to restrain their normal functioning from any malicious attack. Software should be self-defensive by functioning proactively. The best possible way to achieve the security in the software
Measuring the Impact of Software Design on Software Security

Chapter-1: Introduction

is by incorporating it from the commencement of various phases of software development life cycle (SDLC). Security basically concerns with: safety, confidentiality and reliability of the data. The “Security Enhancement” of SDLC activities consists of: a) Existing SDLC activities with practises and checkpoints. b) Adding new activities, practices and checkpoints at few instances. c) Also, it’s recommended to eliminate or fully replace certain activities or practises that are obstructing the ability to produce secured software. So, secured software should be one that does not compromise at any point of time due to any intentional or unintentional threat (S Chandra & et al. 2009). Based on this need many processes were introduced like OWASP’s CLASP, Microsoft’s SDL and McGraw’s touch points for the development of secure software. But still we have not accomplished this task completely. Generally, intruders take the advantage of software and hardware flaws. As a result, they can easily exploit the software by breaking the availability, integrity, confidentiality and non repudiation as well as other aspects of software security (S Chandra & et al. 2009).

1.1 Importance of Design phase in security Estimation:

The software architecture is readily available in the design phase. Hence, it is the best phase to deduce the security of that level of software. Thus, we can reduce the various software flaws which will take care of the product security goals and sensitive information (S Chandra & et al. 2009). Software design helps the designer and the user to communicate with the act of software implementation. Design is an important stage as it predicts the whole software life cycle. It’s not only vital for the development of software but also for redeveloping legacy system (Peterson, 2004). For a defect-free design, it is simple to implement a new module, or change an existing one as well as maintaining it. But this process requires a lot of time and resources as both detection and correction of design defects is to be carried out. Also, this process can have errors as they cross cut classes and methods whose descriptions were mistaken. For such difficulties, many techniques have been proposed by some researchers. These processes include manual inspections which are not suitable for large programs. The detection and correction of design defects was affected by the vagueness of the descriptions based on text, deficient and incorrect structured representation of design defects. But if the design defects were eliminated in the same stage and the design was checked against the security flaws in the same stage, then it
could dismiss the propagation of flaws to further stages. Object oriented technology is popular in industrial software development because it provides higher quality and lower maintenance cost. It is easily possible to assess and evaluate the Object oriented software in its preliminary stages. It requires significant efforts to identify classes, attributes, functions and communication between the classes. The result of this initial assessment is the blueprint for the final implementation. Evaluating the design properties at this stage not only ensures the compliance of the components as per the security contract but also is useful in making selection of on-shelf security components.

1.2 Motivation:
Software development aims at providing secure products with high quality which can not only correct but also steady and comprehensive. The same goes for security. For controlling security, it is necessary to evaluate the results that describe the strengths and weaknesses of the software. In case of object oriented methodology, the quality of software is highly dependent on its design. In literature, many metrics and models were already proposed to evaluate the quality of object oriented software from its design. Security is one of the major factors of quality. Therefore, there is possibility to quantify security based on quality quantification methodologies for object oriented software. We can measure the quantitative properties of a quality attribute at various levels of abstraction. For instance, we can measure complexity: a) at code level (McCabe Cyclomatic complexity), b) at design level (coupling between objects,), c) at architectural level (coupling between components). Thus, security properties can also exhibit a similar behaviour. In object-oriented design, the relationships between components are most essential. Analysis of these relations shows the robustness of design decisions. But there are no existing methodologies or frameworks which measure security in this stage and hence which increase the complexity and cost of security implementation in later stages. So, an approach is required to provide efficient and effective security implementation and assessment at design stage.

1.3 Problem:
The problem definition is “measuring impact of software design on software security” that means measure the strength and weakness of the design from security point of view. Many researchers have identified this problem and made significant
contribution on the measuring impact of object oriented design properties on security but the currently available research work was limited with one or two design properties at the same time and unable to provide the solution for checking the complete design. In this research work an attempt was made to overcome all these limitations by measuring the impact of all important design properties on security.

1.4 Research and Development Challenge:

Security is all about dealing with threats, which normally associated with human factors. We need to consider user behaviour to analyse the efficiency of such property which cannot be seized by figures. That’s why; the assessment of the quality through software metrics does not fulfil all facets of security. Software susceptibility relates to software flaws. A flaw is defined as an accidental situation that hinders the general performance of the functional unit. The results of software vulnerabilities may differ than what is expected in fault prediction, although susceptibilities are subset of faults. This happens because of the characteristic differences in the faulty code and the vulnerable code have not been investigated quantitatively yet (Riccardo Scandariato, et al 2006).

As there is no standard terminology or methodology, many measures were in ambiguity which limit their use and make it difficult to relate to one another. For example, there are different decisions made while defining a coupling measure, these decisions have been made with respect to the goal and by defining an empirical model based on clearly stated hypothesis. Literature has gaps in clearly defining the properties which should be considered for security (Lionel C. Briand et al 1999).

Many researchers have made contribution on measuring impact of object oriented design properties on security but their research work is limited and is not providing the solution for checking the complete design.

In addition to this, software security is dynamic in nature. Software that is protected in a specific atmosphere with a precise risk may no longer continue to be safe if the software or the environment itself is changed or modified.

Software testers can course out 10,000 hours of testing and eventually be very assured that the software that clears those checks will function consistently but the same cannot be supposed for the security of the software. The security testing techniques haven’t been developed completely yet and they collected, form a partial coverage of all issues related to security testing (Thomas R. Peltier, et al, 2005).
Hence, our research work attempt to consider all the limitations as stated above and will evaluate the strength and weaknesses of design.

1.5 Overview of the Solution:
Developing secured software not only has many advantages but has now become a need for software organization that has three main properties (S. Chandra and R. A. Khan, 2008):

• **Dependability**: Under all conditions dependable software executes predictably and operates properly including hostile conditions and even when software runs a malicious host or comes under any attack.

• **Trustworthiness**: Trustworthy software is the one that contains less weaknesses and vulnerabilities that can be intentionally exploited to subvert or sabotage the software’s dependability. It must also not contain any malicious code which will cause it to behave in a malevolent way.

• **Survivability**: Survivability is also referred to as “Resilience”. Survivable software is the one that is resilient enough to: resist or tolerate most known attacks and as many novel attacks as possible (i.e. protect itself against attacks and continue operating). Recover as quickly as possible and with little damage as possible from those attacks that it can neither resist nor tolerate.

Considering above aims, an attempt was made to investigate methods for systems that support effective study and design at the software architecture level, where the structure emphasis on the requirement of a group of components and their performance, networks among constituents, and constrictions of the system. The sensitive information, special attention needs to be given in the designing phase for keeping the information confidential. To achieve this goal, detailed knowledge of the causes and prevention of vulnerabilities is essential.

Software security is accomplished by introducing the concept of “secure by design” approach. In software engineering, “Secure by design” refers to the point that the software has been premeditated in such a way that it is secure from the bottom level. Malicious exercises are not considered with concern until and unless a security vulnerability is discovered or an invalid user input is detected. This approach is not covering the application level security or emerging security polices in software life cycle. Focus is only on the design of software, testing the security of design and providing necessary data for designer to develop secure design.
1.6 Objectives, Goals and Scope of Research Work:
The objective of research work is to measure the impact of software design on software security. In the proposed work, we are measuring the impact of all important design properties on security by developing the security model for object-oriented design (SMOOD). This model can assess the security in design phase, which in-turn allows the designer to do the necessary action to deliver the secure software, which is secure by design. This helps in diminishing the development time as well as the cost by detecting and eliminating the defects in the initial stages (Dai, L, 2006). The traditional methodology for security estimation is not applicable for object oriented software. Software engineers, trained in object-oriented technology, need to recognize the desirable and non-desirable properties of systems, and their effect on the security factor. The desirable properties should ultimately lead to more available, confidential and integrated software products. To relate the desirable and non-desirable properties, there must be some formal definition or mechanism to detect the presence of these properties. Hence, the above information is highly needed for analysing the design security in consideration to structure, classes and associations prior to their implementation.

Software security is a combination of internal attributes and external attributes. External attributes includes availability, confidentiality and integrity. Whereas internal attributes consist of the design properties. External attributes can be easily expressed but are hard to measure whereas the internal attributes are hard to express but easy to measure. It has been proven that by improving internal attributes, external attributes get affected. So, there is a need to fill the gap between internal attributes and external attributes. In this work, we have tried to overcome this gap. Based on this concept, quantification of security at design stage can be done using security attributes and design properties.

The Scope of this research work is allied with the design properties of object oriented methodologies, which are reliably evaluated through analysing the internal as well as external assembly, association and functionality of methods, attributes, design components and classes. Valuation of a class definition by specifying its attributes and methods and relationship with other classes unveils noteworthy information that neutrally captures the functional and structural features of a class and its objects. Some design properties are same for structural and object-oriented development like
abstraction, encapsulation, coupling, cohesion, complexity and design size, and some additional design properties for object-oriented development like: inheritance, messaging, polymorphism, and composition as well as class hierarchies (Jagdish Bansia 2002). All these properties were considered in the present research work.

1.7 Benefits of Proposed Methodology:
The benefits of proposed security model for object oriented design (SMOOD) were summarised as follows:

Testing security of design before building the software: Design testing was usually cheaper than building a complete system then tests it. And this helps to test system to identify flaws to be corrected at initial stage.

Choice can be made from multiple designs: By providing imperial data of design properties it is possible for designer to choose secure design from multiple designs.

Reduction of cost: Due to consideration of security from earlier phase of software development, cost of software could be reduced.

Creativity: The simplicity of creating and modifying designs helps system developer to experiment with creative thought and innovation at little cost. Multiple architectures and design solutions can be tested easily before writing code.

1.8 Thesis Organization:
The thesis starts with title pages, index, list of tables, list of figures and abstract. There are about nine chapters for this thesis based on research work. At the end of thesis, there is bibliography related to research work.

Chapter 1: Introduction:
This chapter briefly discusses about the needs of secure development and secure modelling. It provides the objectives, goals and detailed scope for carrying out the said study. Problem is well defined with overview of solution. How this research work can be made a challenging task was mentioned in research and development challenges. Benefits of proposed modelling methodology were also described briefly. It concludes with organizational aspects of thesis.

Chapter 2: Background and Literature survey:
This broad chapter is divided into various backgrounds like basic concepts, definitions, background terminology related to security modelling technologies which can be used and have been referred in our research work. The review of literature
survey describes the comprehensive literature survey which can review the different problem domains related to secure system development models, security approach, security consideration at design stage, vulnerability identification and analysis, threat etc. The significant conclusions derived from literature review were summarized from more than 100 research papers and finally, the limitations of existing security approaches and methodologies were discussed.

Chapter 3: Approach:
This chapter gives an approach towards the research work carried out related to proposed security model for object oriented design (SMOOD).

Chapter 4: Requirement Engineering:
In this chapter a detail requirement related to security model SMOOD were listed and based on that software requirement specification (SRS) was prepared.

Chapter 5: System Analysis:
In this chapter, identifying the data and the method used to deduce solution to the problem were focused. Also, the collection of information about security concept, security principals, security attributes and object oriented methodology with design properties have been taken care of.

Chapter 6: System Modeling:
These chapters demonstrate four case studies- library management system, bank management system, hospital management system and courseware management system. Multiple designs of each case study were developed with respect to changes in design properties and calculations of security metrics and measurement of design properties on each diagram of case studies were reported.

Chapter 7: System Design:
In the chapter seventh, the result which has been collected in chapter 6 and also the data collected in chapter 5 were used to develop security model for object oriented design (SMOOD).

Chapter 8: System testing for security Model:
This chapter covers testing of different commercial open source software with proposed security model (SMOOD) and analysis of obtained results.

Chapter 9: Conclusion:
Chapters from 1 to 9 were summarised. Then survey of contributions made by this research work was outlined. Finally, this chapter concludes with recommendations for continuity of research in the said area was given.
Bibliography: This chapter covers referred and used research papers, books, websites, and reference manuals for the thesis.

1.9 Discussion:
Secure software is a robust one. The dependability of the software remains unchallenged. In comparison to other software properties, the perspective of security in software is not well comprehended. The user’s or developers are not able to regulate with 100% surety the security of software in its development life cycle. Presently, there was no efficient method available that constitutes a security assessment. There was a need to design standards that were flexible in design to accommodate a range of security assurance levels. If software is protected by design it is very inflexible for impostors or external intrusions to breakdown the software. In earlier research work this type of model was available for checking quality of design. Thus by measuring the impact of design properties on software security, a new security model for object oriented design (SMOOD) has been developed in this research work which was concerned with system level security. With this security model we can verify the design security such that if several designs were presented then designer can definitely select the protected design or if the designer discovers that security of present design are low then he can do required alteration in the design stage.