SOFTWARE SECURITY PATTERNS IN SECURITY ENGINEERING

Rshma Chawla*

Dr. Naveeta Mehta**

ABSTRACT

Secure software is the essential need of the time. Security has become a important feature of any software system. Security issues are always the secondary task for the developers in SDLC process. But in accordance with current scenario security should be given the highest priority in all phases of SDLC. The major problem faced by developers is unavailability of information about recent attacks and the way to cure them. This paper elaborates Security patterns as reusable solutions for security related problems in SDLC as even till date security comes after the development phase. Why it is not included in each phase? Better quality specification can be produce at lower cost by indulging generalization, reusable security requirements.

Keywords: *Security, Patterns, Design Patterns, Security Patterns.*

*Lecturer, M.M. Institute of Computer Technology & Business Management, M.M. University, Mullana, India.

**Associate Professor, M.M. Institute of Computer Technology & Business Management, M.M. University, Mullana, India.

1.0 INTRODUCTION

Faults created by software developers open defect in applications and expose their vulnerabilities. Give security expertise to software developers is difficult .In SDLC security expertise is one frequent missing quality that needs to be addressed strongly, by taking advantage of the scaling effect of security patterns[1]. Security patterns capture security experts' knowledge for a given security problem. Therefore security patterns are developed by security experts and are used by as software developers. Various standardized methodologies were developed to for secure system development. Because security experts are again required to interpreter such standards. The successful utilization of security expertise in security patterns is a fundamental added value towards providing security for non-security experts [2].

A Security Pattern is a description of one particular recurring security problem that arises in specific contexts and presents a well-proven generic scheme for its solution [13, 14].The software developers can be either have coding skills for software development or designer skills for taking responsibility of designing several programming[1].

The rest of the paper is organized as section 2 describes that where Security Engineering meets Security Patterns. Section 3 give the overview of reusability approach for patterns deployment in various phases of SDLC and finally section 4 conclude the paper .

2.0 SECURITY ENGINEERING MEETS SECURITY PATTERNS

Tackling with the challenges of developing secure software systems remains a creative research area in security engineering. Numbers of security methods have been developed to assist developer for establishing and maintaining security in software development. These methods differ regarding their structure, granularity, flexibility, usability, costs, or whether they are formal or informal [15] such as Security Policy, Evaluation Criteria, Tree Representations, Formal Methods, Semi-formal Approaches. At present documentation of recurring security problems can be done as security patterns. Security patterns provide summarized solutions to specific security problems and can be used to build secure systems by designers with little knowledge of security. A Security Pattern System gives association between various Security Patterns. Security Patterns have several advantages: Novices can act as security experts, security experts can identify, name and discuss both problems and solutions more efficiently, problems are solved in a structured way, and dependencies of components can be identified and considered appropriately [15]. Security engineering could be done more efficiently if developer gets the help for managing various security aspects.

Software patterns are structured into many sections that's make them capable of handling different aspects such as the pattern name, problem and the solution. The more information a pattern has, the more important structure becomes [4]. Structure leads to uniformity of patterns. Templates can be designed according the authors ideology but the main purpose of pattern must be maintained. Many approach to security patterns are given by many authors. There are different pattern based approaches to the existing security problems. For example, E. B. Fernandez [6] has collection of security patterns which was using UML diagrams. Scumacher and Roedig [8] propose the use of patterns in security engineering with no specific template. Kienzle et al[5] developed tutorial for writing security patterns. Yoder and Barcalow[7] has collection of security patterns without any diagrams. There are different security pattern templates depending on authors. According to author [5] a security pattern template consists of the following elements: pattern name, abstract, problem, solution, issues, examples, trade-offs, related patterns, references. Template shown by David et.al [9], have the following sections: intent, context, problem, description, solution, consequences, trade-offs and results. Whereas author[4] describe the elements of security pattern template as: applicability, behaviour, constraints, consequences, related security patterns, supported principles. However [10] Angel Cuevas the elements of the pattern are problem/requirements and context, solution, pre-conditions, properties, features, consequences. The description of these basic elements as a whole is given below:

- **Pattern Name:** The Pattern Name should capture the essence of the pattern in a concise and, if possible, catchy manner.
- **Intent:** It describes what the pattern does, which its rationale and intent are, and what particular design issue it addresses.
- **Context:** It describe the context of the problem
- **Aliases:** The Aliases should enumerate other names for the pattern, including names by which others might have referenced the pattern in the literature.
- **Implementation Issues:** The Implementation Issues should provide some wisdom in the form of detailed hints and techniques.
- **Abstract:** The Abstract should summarize the pattern briefly in two to three sentences. The summary should include what the purpose or intent of the pattern is.
- **Problem:** The Problem should describe the conditions that motivate the usage of the pattern.

- **Solution:** The Solution should describe at a high level how the pattern solves the problem described in the problem statement.
- **Dynamic Structure:** The Dynamic Structure should outline the interactions between the various components in the static structure.
- **Common Attacks:** The Common Attacks should identify attacks that interact with this pattern.
- **Sample Code:** Developers appreciate sample code that they can link directly into their application and begin using immediately.
- **Consequences:** The Consequences should describe the possible impact of using the pattern with respect to various functional and non-functional requirements.
- **Applicability:** The applicability field is important in determining if a pattern is applicable to a system.
- **Known Uses:** The Known Uses should cite examples of this pattern that are known to be in actual use.

3.0 REUSABILITY OF SECURITY PATTERNS

Security requirements engineering show possibility for reusing security requirements and goals, which reduces the overall expenditure cost of requirements engineering and gives quality of requirements specifications. Software designers and security professionals after making security pattern include it in the repository for reuse. So that it can be further used by other designers in their software requirement phase. As a result, designers can rapidly model collections of interrelate security capabilities without introducing the risks and problems.

Reuse of security patterns provides several advantages to the developers to produce secure software with less no of vulnerabilities.

- Opportunity: Many systems face similar security threats and deal with them in the same standardized ways. This is an opportunity to define common security measures and establish reusable security pattern for future projects [11].
- Reduced cost: Reuse reduces the effort needed to produce security pattern for later projects. Writing security pattern that can be reused is a time investment in future productivity [12].
- Improved quality: A security pattern written particularly for reuse will have been given proper attention and inspected for quality. Reusing published pattern thus results in fewer defects [11].

Reusing predefined does have its disadvantages.

- There is an up-front investment in effort and resources associated with creating a reusable security repository.
- Writing goals and requirements for reuse may take some extra thought, and cataloging them for future use will require additional process management.

Reusability is unlikely to provide any immediate advantage when it is first adopted. [12]

4.0 CONCLUSION

This paper covers various security issues in SLC and describe that how reusability mechanism can be useful in SDLC with the help of security patterns for developing secure systems are discussed. Security pattern are reusable solution to the frequently occurring problems .It therefore helps developer to develop error free software. Yet it is not possible to develop fully secure system and still there is a big gap between the availability, production and utilization of security patterns in various phases of software development. In future more and more security patters need to develop and developers must understand the need of security parameters throughout software development process.

5.0 REFERENCES

1. P. El Khoury, A.Mokhtari "An Ontological Interface for Software Developers to Select Security Patterns" 19th International Conference on Database and Expert Systems Application © 2008 IEEE
2. J. J¨urjens, G. Popp, and G. Wimmel. "Towards Using Security Patterns in Model-based System Development". In Euro-PLoP 2002 (Security Focus Group), 2002.
3. J. Vlissides. Seven Habits of Successful Pattern Writers. C++ Report, 1996. SIGS Publication.
4. A.Van Lamsweerde. "Goal-oriented requirements engineering:A guided tour". In RE01 - 5th Intl. Symp. Requirements Engineering, pp 249–263. IEEE, 2001.\
5. IBM Global Technology Services. IBM Internet Security Systems X-Force Trend Statistics, 2008. http://www-935.ibm.com/services/us/iss/xforce/midyearreport/ xforce-midyear-report-2008.pdf
6. Feiertag R.J., Levitt K.N. , Robinson L., "Proving Multilevel Security of a System Design", Proc. Of the sixth ACM symposium on operating systems principles,New York, USA, 1977.
7. Gervasio, Alejandro. "Validating User Input with the Strategy Pattern." Developer Shed, March6, 2007. http://www.devshed.com/c/a/PHP/Validating-User-Input-with-the-Strategy-Pattern/

8. Microsoft Developer Network. “Input Sample: Demonstrates User Input Validation on Client and Server.” MSDN Visual Studio Developer Center. http://msdn.microsoft.com/enus/library/x88c4k6b(VS.71).aspx (2009).
9. K.Suresh Babu , Dr.K.Chandrasekharaiah “Security Patterns: State-of-the-Art Scenario” IJCSNS International Journal of Computer Science and Network Security, VOL.11 No.4, April 2011,pp-131-135
10. Viega J., “Security in the software development life cycle:an introduction to CLASP, the comprehensive lightweight application security process”, IBM, 2004.
11. M. Schumacher, "Security patterns and security standards",EuroPLoP 2003.
12. J. Yoder and J. Barcalow, “Architecural patterns for enabling application security,” in PLoP97 Conference,1997.
13. M. Schumacher. “Security Engineering with Patterns - Origins,Theoretical Models, and New Applications”, volume 2754 of Lecture Notes in Computer Science. Springer, 2003.
14. M. Schumacher, E. Fernandez-Buglioni, D. Hybertson,F. Buschmann, and P. Sommerlad. Security Patterns : Integrating Security and Systems Engineering (Wiley Software Patterns Series). John Wiley & Sons, March 2006.
15. Markus Schumacher and Utz Roedig “Security Engineering with Patterns“, Markus Schumacher. PLoP 2001 conference ,2001.