Security Patterns: State-of-the-Art Scenario

K.Suresh Babu[†] *, Dr.K.Chandrasekharaiah*[††]

[†]Research Scholar, School of IT, JNT University Hyderabad, India

[††]*Prof. in CSE, School of IT, JNT* University Hyderabad, India

Abstract
A Pattern is a packaged reusable solution to a recurrent problem. Design patterns are based on the expertise and knowledge of the software scientists in their particular area. Security patterns are intended to capture security expertise in the form of worked solutions to recurring problems. In this paper we have made study on a range of recent work on security pattern, their features, template approaches, and their classification. This also deals with usage of security patterns to the several fields of computer science.

Keywords:
patterns, cataloging, expertise, knowledge

1. Introduction

In the life cycle of a software development process, the role of security has been increasing rapidly. In each and every aspect of software development we need to think of the security parameters. Security has become an important aspect for the software systems. But effectively addressing the security problems is difficult because traditional software development life cycles do not deal with security concerns at all. There is no structured guidance on how to design security into the engineering process, and "security" by itself is not a feature that "demos well [17]." Security problems in software have become common such that it has become very difficult to identify and remove them, and the level of vulnerability continues to increase: The Carnegie Mellon university's Computer Emergency Response Team(CERT) Coordination Center reported approximately 7,236 vulnerabilities in 2007[16], an increase over the past years. There is a huge disconnect between security professionals and systems developers. Security professionals are primarily concerned with the security of a system, while developers are primarily concerned with building a system that works.

While security is one of the nonfunctional goals that developers must be concerned with, it is but one of many. And while security professionals complain that developers don't take security seriously, developers are just as frustrated that security professionals don't understand that security is not their only concern. Hence many models came in to sight like, The Formal Security Model [18], Waterfall Model with Security [19], comprehensive Lightweight Application Security Process(CLASP) [20], which emphasizes the security concepts in software development life cycle.

Security concerns and issues have become more and more crucial in the software development process. According to [1], security concerns must inform every phase of software development, from requirements engineering to design, implementation, testing and deployment.

Hence, Security patterns are proposed as a means of bridging this gap. Security patterns are intended to capture security expertise in the form of worked solutions to recurring problems. Security patterns are intended to be used and understood by developers who are not security professionals. While the emphasis is on security, these patterns capture the strengths and weaknesses of different approaches in order to allow developers to make informed trade-off decisions between security and other goals.

Finally, security patterns are software patterns for the security services or goals. The development of security patterns is depicted in Figure 1, as the combining of software patterns and the security services that gives rise to security patterns.

Figure 1. Development of Security Patterns

The remainder of this paper is organized as follows. Section 2, about the features of security patterns, section 3 deals with security pattern template, section 4 discuss about cataloging of security patterns, section 5 deals with usage of security patterns and finally section 6 talks about the conclusion and future enhancement.

2. Features of Security Patterns

A security pattern is a well-understood solution to a recurring information security problem. A security pattern encapsulates security expertise in the form of worked-out solutions to the recurring problems, presenting issues and trade-offs in the usage of the

Manuscript received April 5, 2011
Manuscript revised April 20, 2011

pattern. Above all, security patterns are meant to be constructive. It provide all the available security expertise in the list form to the developers with respect to what to do and what not to do. A mediocre developer who attempts to understand security is likely to be overwhelmed by these lists. Security patterns instead try to provide constructive assistance in the form of worked solutions and the guidance to apply them properly.

Before knowing the features of security patterns we should be well aware of security services. Security services [2] include authentication, access control, data confidentiality, data integrity, nonrepudiation and availability.

The authentication service is concerned with assuring that a communication is authentic. RFC2828 and X.800 define two specific authentication services: Peer entity authentication and Data origin authentication. Access control is the ability to limit and control the access to host systems and applications via communications links. Confidentiality is the protection of transmitted data from passive attacks. Confidentiality is the protection of the traffic flow from malicious analysis. This requires that an attacker be not able to observe the source and destination, frequency, length or other characteristics of the traffic on a communication facility. As with confidentiality, integrity can apply to a stream of messages, a single message or selected fields within a message. Nonrepudiation prevents either sender or receiver from denying a transmitted message. Thus, when a message is sent, the receiver can prove that the alleged sender, in fact, sent the message. Similarly, when a message is received, the sender can prove that the alleged receiver, in fact, received the message. While designing the security patterns we have to see that one or more of the above security services are implemented. In other words, security patterns refer to applying the security services to the software patterns. The security patterns thus provide all the security services in various forms based on the proven generic solutions.

3. Security Pattern Template : Different Approaches

Generally, software patterns are organized into multiple sections so that they can handle different aspects such as the pattern name, problem and the solution. Templates can be designed according the authors ideology but the main theme of the pattern should be maintained. Different authors have different approach to security patterns.

There are different pattern based approaches to the existing security problems. For example, E. B. Fernandez [15] has collection of security patterns which was using UML diagrams. Scumacher and Roedig [25] propose the use of patterns in security engineering with no specific template. Kienzle et al[3] developed tutorial for writing security patterns. Yoder and Barcalow[23] has collection of security patterns without any diagrams. There are different security pattern templates depending on authors viewer point[39]. Here we try to survey the various security patterns templates.

According to, [3], a security pattern template consists of the following elements: pattern name, abstract, problem, solution, issues, examples, trade-offs, related patterns, references. In [4], has the following sections: intent, context, problem, description, solution, consequences, trade-offs and results. In [5], the elements of security pattern template are given otherwise as : applicability, behavior, constraints, consequences, related security patterns, supported principles. In [6] Angel Cuevas et al., the elements of the pattern are problem/requirements and context, solution, pre-conditions, properties, features, consequences. The elements of various security pattern approaches [3, 4, 5, and 6] are presented in Table 1.

Table 1. Elements of Security Patterns indicating differential author styles

Author Approaches/ Elements	Darrell M.Kienzle [3]	David G. Rosado [4]	Betty H.C.Cheng [5]	Angel Cuevas[6]
Pattern Name	X	X	X	X
Intent		X		X
Context		X		X
Abstract	X	X		
Aliases				X
Problem	X	X	X	X
Solution	X	X		X
Static Structure			X	
Dynamic Structure			X	
Implementation Issues	X			
Common Attacks				X
Known Uses	X			X
Sample Code	X			
Consequences		X		X
Applicability	X	X	X	
Constraints		X		
Supported Principles			X	

Based on our survey, the different elements of security patterns are considered as pattern name, intent, context, abstract, aliases, problem, solution, static structure, dynamic structure, implementation issues, common attacks, known uses, sample code, consequences, applicability, and constraints. The description of these different elements is as follows

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

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.

Aliases: The *Aliases* should enumerate other names for the pattern, including names by which others might have referenced the pattern in the literature.

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.

Static Structure: The *Static Structure* should present the constituent elements involved in the usage of this pattern. **Dynamic Structure:**The *Dynamic Structure* should outline the interactions between the various components in the static structure.

Implementation Issues: The *Implementation Issues* should provide some wisdom in the form of detailed hints and techniques.

Common Attacks: The *Common Attacks* should identify attacks that interact with this pattern.

Known Uses: The *Known Uses* should cite examples of this pattern that are known to be in actual use.

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.

Constraints: The constraints field contains global conditions that have to apply in order for the security pattern to achieve its intended goal.

The design pattern template provided by Gamma et al [26] has got some pre-defined elements. These design pattern templates can be extended and modified to get the security pattern templates to fit our application needs. The extension and modification has been depicted in the Figure 2.

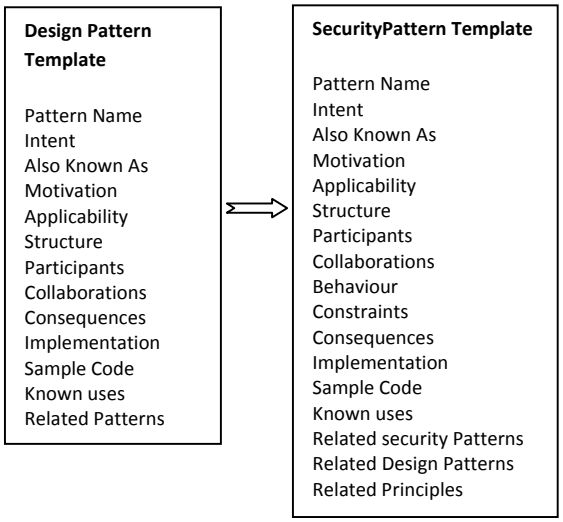

Figure 2. Extension of Design Pattern Template to Security Pattern Template

4. Security Patterns Classification

Mathematically, classification is performed as a tree based mechanism, where the root node provides a more general concept that is refined by each descendant. Another classification method is to partition of the domain in discrete groups based on some criteria. The security patterns classification efforts are based on partitioning the systems space. Here we list a survey of the classification schemes [7]. The different classification schemes are based on criteria such as (i) applicability, example patterns are VoIP Tunnelling, Network Segmentation, Secure VoIP Call, Signed Authenticated Call, etc.[21] (ii) product and process, example patterns are Security Patterns for VoIP [21] (iii) logical tiers, example patterns are Secure Communication, Secure Association, etc. [22] (iv) security concepts, example patterns Single Access Point, Check Point, etc [24] (v)system viewpoints and interrogatives, example patterns are Full View with errors, Limited View, etc [23] and are presented in table 2.

The security pattern repository share[8] in the patternshare website provides a comprehensive listing of security patterns from different sources. The McCumber cube [9] mechanism examines security in the context of

information state. An analyst using the McCumber cude identifies the information flow of a system, parses the information flow for security-relevalant environment and then maps the findings on a rubiks cube like structure. In the Cube the X-axis represents the three primary categories of safeguards, i.e., technology, policy and procedure and human factor. The Y-axis of the model represents information states of transmission, storage and processing. The vertical axis comprises the three security perspectives of confidentiality, integrity and availability. I4 security patterns have been defined based on the classification of security patterns based on the McCumber cube. The classification scheme based on application context[7] considers the structure of the system and partitions the patterns based on which part of the system they are trying to secure. Based on this 14 security patterns have been defined. A security wheel scheme represent the security features as the spokes in a wheel[10]. At the core of the hub of the wheel is the service or application that is under consideration. The spokes represent 12 core security services applicable to the service. The edge of the wheel represent perimeter security. The CIA model[11] scheme discusses the key issues of security i.e., confidentiality, integrity and availability. It defines 14 security patterns. The concepts of STRIDE[12, 13] which is based on threat modeling, is used to classify the patterns and developed 12 patterns.

Table 2. Classification Criteria for Security Patterns

S.No.	Classification based on
I	Applicability
Ii	Product and process
Iii	Logical tiers
Iv	Security concepts
V	System view points and interrogatives

Similar to Design Patterns [26], the security patterns can be organized by purpose in to structural, behavioral and creational patterns. They can also be organized based on abstraction levels like, application level(objects are deployed at client), host level(objects running at server), network level(objects are distributed over the network).

5. Usage of Security patterns

Security patterns can and should be applied to develop secure systems. These security patterns help the systems to be more secured. The patterns should be applied at each and every stage of software development life cycle. The basic theme of security pattern [15] is that security principles should be applied at every stage of the software lifecycle and that each stage can be tested for compliance with the security requirements. They should be implemented [14] at domain analysis stage, requirements stage, analysis stage, design stage and implementation stage. Security patterns are used for modeling and analyzing secure systems.

Process for Using Security Patterns

We demonstrate how this process can be applied to a simple ecommerce system.[5]

1. Use the security patterns to construct a basic foundation for the system and refine the UML high-level structural and behavioral models of the system
2. Perform consistency checks and generate a role model of the system from the UML diagrams.
3. Using visualization utilities to display simulator output within the context of the UML diagrams, simulate the model from the first step for validation purposes, and then refine the system model if errors are discovered.
4. Specify properties for the system by instantiating th specification patterns from the Constraints field of security patterns. Determine which parts of the system are relevant to a given property to bechecked, and introduce abstractions for the remaining parts of the system where possible. To simplify analysis, create equivalence classes for attribute values when possible.
5. Use a model checker to determine if the properties are satisfied. If errors are detected, then use visualization utilities to display the counterexample in terms of the UML diagrams, and then refine the system model.

6. Conclusion and Future Work

In this paper, an account of classification of Security Patterns as per the existing literature is presented. In view of the diversity of approaches by patterns researchers, an overall perspective representing detailing the feature support and templates with regard to the Security Pattern explored is brought out. The Classification and usage of Security Patterns is being studied. Mobile service applications have the potential as an area for devising Security Patterns towards future research.

References

[1] P.T. Devanbu and S.Stubblebine, "Software Engineering for Security: A Roadmap" in Proc. Of the Conf on The Future of Software Engg, pp.227-239, 2000.
[2] William Stallings, " Cryptography and Network Security", Fourth Edition, Pearson Education, 2009.
[3] D.M.Kienzle and M.C. Elder: Security Patterns for Web Application Development, Final Technical Report, Univ. of Virginia., 2002.
[4] David G. Rosado, Eduardo Fernandez-Medina, mario Piattini, "Comparision of security patterns", IJCSNS International Journal of Com. Sci & Net Sec, Vol.6 No.2B, Feb 2006.
[5] Betty H.C. Chengy, Sascha Konrad, Laura A. Campbell, and Ronald Wassermann, "Using Security Patterns to Model and Analyze Security Requirements", Proc.

International workshop on requirements for high assurance systems, 2003.
[6] E.Houg, N.R.Mead and T.R.Stehney, "Security Quality Requirements Engg.(SQUARE)methodology", Technical Report CMU/SEI-2005-TR-009, CMU/SEI, 2005.
[7] Munawar Hafiz, Ralph E. Johnson, "Security Patterns and their Classification Schemes", PLoP 2006.
[8] Security Pattern Repository, www.patternshare.com
[9] J.R.McCumber Information systems security: A Comprehensive model. In Proc 14th NIST-NCSC National Computer Security Conference, Washington, pages 328-337, October 1991.
[10] "CIA Model", Security Conference, Washington, pages 328-337, October 1991.
[11] Commission of European Communities. Information technology security evaluation criteria, version 1.2, 1991.
[12] M. Weiss, H. Mouratidis, "Selecting Security Patterns that Fulfill Security Requirements", Proc. 16th IEEE Intl. Requirements Engg. Conf. 2008.
[13] Lauri Kiiski, "Security Patterns in Web Applications", TKK T-110.5290 Seminar on Network Security, Helsinki University of Technology, Oct 2007.
[14] E.B.Fernandez, N.Yoshioka, H Washizaki, J Jurjens, M. Vanhilst, G Pernul, "Using security patterns to develop secure systems", Ist International workshop on Software Patterns and Quality(SPAQu'07), Nayoga, Japan, Dec 2007.
[15] E.B.Fernandez, N.Yoshioka, H Washizaki, J Jurjens,"Using security patterns to build secure systems", Ist International workshop on Software Patterns and Quality(SPAQu'07), Nayoga, Japan, Dec 2007.
[16] CERT statistics, 2008. http://www.cert.org/stats/cert_stats.html
[17] Richard Sinn, " Software Security: Theory, programming and Practice", Cengage Learning, 2008.
[18] 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.
[19] McGraw G, "Software Security", IEEE computer society digital library, 2004.
[20] Viega J., "Security in the software development life cycle: an introduction to CLASP, the comprehensive lightweight application security process", IBM, 2004.
[21] E.B.Fernandez, J.C.Pelae, and M.M.Larrondo Petrie, "Security Patterns for Voice Over IP Networks", Journal of Software, Vol. 2 No. 2, August 2007.
[22] The Open Group "Guide to Security Patterns", The Open Group , April 5, 2002.
[23] Joseph Yoder and Jeffrey Barcalow, "Architectural Patterns for enabling Application Security", PLoP Conf., 1997.
[24] Ronald Wassermann and Betty H.C. Cheng, "Security Patterns", Michigan State University, PLoP Conf., August 2003.
[25] M. Schumacher, "Security patterns and security standards", EuroPLoP 2003.
[26] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software", Addison-Wesley, 1994.