Introduction to Agile Security Architecture Model

M.upendra Kumar

Associate Professor CSE MGIT Hyderabad India

Abstract — In this paper, Introduction to paper title is presented, with some main concepts like Software Engineering, Security Engineering, Software Architecture, Security Architecture, Layered Pattern, Model Driven Architecture (MDA), Agile Software Development, Designing Solutions, Dependable Privacy Requirements, Next Generation Secure Web Engineering Applications Case Study using Agile Modelling, Web Services, and Web 2.0. Finally, motivation of the paper, organization of the paper is provided in this paper.

Keywords — *Security Architecture, Agile Modelling*

I. INTRODUCTION

Software Engineering

Software Engineering is the application of Engineering Principles to Software Development [1]. Engineering Principles are very practical and pragmatic ones, which will always work or will never work at all. It guarantees any Software Application Development project success to be developed with in time and within budget [2]. It generally has phases like Software Requirements, Software Architecture Design, Software Development (or Coding or Implementation), Software Testing, Software Maintenance etc [3]. It has traditionally various theories like philosophy, cognitive informatics, denotation mathematics, system science, organization laws, and engineering economics. There is no silver bullet in Software Engineering Design [4], i.e. General solution for any specific problem has no silver bullet solution. Software Engineering profession requires a true discipline, in terms of design of architectures, coding principles etc. Software Engineering research examines critical issues in software engineering including complexity, structure, and evolution of software systems [5]. Software Engineering has developed 2500 programming languages, requires data mining for lost requirements should be a standard practice. Traditional Software Engineering models are: Waterfall model, Spiral model, V-model etc. Now popular models used are agile methods [6].

Secure Software Engineering or Security Engineering

Secure Software Engineering or Security Engineering deals with integration of Security into the phases or layers of Software Engineering process. As Security threats against software increase in numbers and severity, fundamental changes are needed in Software Architecture, Design, Coding Practices, and defensive methods. It is a sub-field of the broader field of computer security. Security generally means CIA (Confidentiality, Integrity and Availability). Confidentiality or Secrecy is achieved using Encryption / Decryption Cryptographic systems. Integrity means information should be unchanged what so ever. Availability means its strategy against resisting Denial of Service attacks. Software Security design principles, methodology, and concrete programming techniques are needed to build secure software systems [12]. Traditional models for Secure Software Development are: Appropriate Effective guidance for Information Security (AEGIS), Secure Software Development model (SSDM), Waterfall-based Software Security Engineering Process Model, Comprehensive Lightweight Application Security process (CLASP), Security Development Life cycle (SDL) McGraw 7 touch points of security development life cycle viz: code review, architectural risk analysis, penetration testing, risk based security tests, abuse cases, security requirements and security operations. Various Security Engineering standards are: ISO/IEC 15408 common Criteria, ISO/IEC 15288:2008, ISO/IEC 27000 series, Systems Security Engineering Capability Maturity Model. Security Engineering is a sub field of computer security, which deals with tools, techniques and methods that support development and maintenance of systems resisting malicious attacks for damaging systems or data. Application security deals with software engineering problem where system is designed to resist attacks. Standard for Security Engineering is Capability Maturity Model (CMM) for System Security Engineering.

1.1 SOFTWARE ARCHITECTURE

Software Architecture (SA) is at the core of the Software Application development, which is developed at the Design phase, based on the earlier Software Requirements Specification (SRS), and this Software Architecture will be taken as the base line for next phase i.e. Coding or Development or Implementation.

SA = {Elements, Form, Rational}

A set of architectural elements (processing, data and connecting), the form of these elements as principles guiding the relationship between the elements and their properties, and finally rationale (design decisions) for choosing elements and their form in certain way.

Software Architecture Process and Architecture Life cycle has phases like Analyzing problem domain, Design and describe architectural decisions, Architectural evaluation, Realize architecture, Maintenance of architecture. Architecturally Significant Requirements are those requirements that have a measurable impact on software systems architecture. Traditionally Software Architecture design methods are using design patterns and architectural styles. Documenting Software Architectures provides various views like Logical view, Process view, Development view,

Physical view, and Scenarios. Traditional Software Architecture evaluation tools are: Software Architecture Analysis method (SAAM), Architecture trade off analysis method (ATAM), Architecture level maintainability analysis (ALMA) and Performance Assessment of Software Architecture (PASA).

1.2 SECURITY ARCHITECTURE

Security can be highlighted as part of the system development life cycle [11]. Security Architectures enables numerous implementations which are resilient to appropriate and broad based spectrum of threats. Research issues here includes: Complexity is the source of security holes; security is the matter of the weakest link; Tradeoffs needs to be based for complexity versus protection, performance, usability and flexibility. Important Security analysis and design issue is: "How well the system authenticates users and provides protection to data and application layers?" Security generally implies CIA that involves Identification (or Authentication), Authorization (access control mechanisms) and Encryption (using Confidentiality) Security Activities involved in Requirements Engineering phase are Security Use case development, Misuse cases, Threat modeling and Risk Management. Security Activities in Secure Design phase involves: Logical Security Architecture, Physical Security Architecture, Component / Service Security architecture, Design rules and Refined Architectural Risk Analysis. Figure 1.1 depicts the various Attacks on Software Architectures. Security attacks as classified under Resisting attacks, or Detecting attacks or Recovering from attacks. This research focuses on resisting attacks in terms of Authentication of Users and / or Authorization of Users.

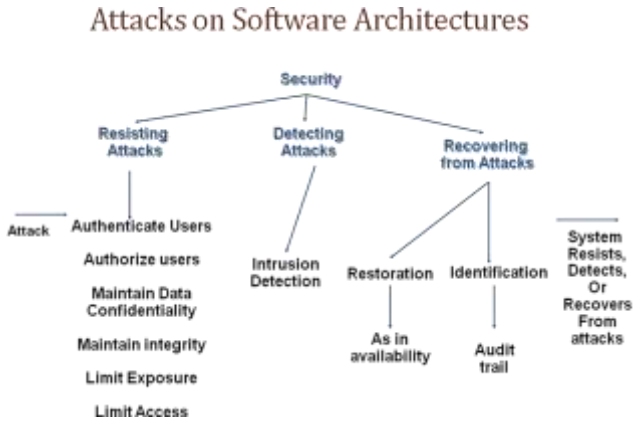

Fig 1: Attacks on Software Architectures

Security Architectures has to be designed and implemented for integrating security requirements of Security Engineering with Software Architecture design [37]. Securing at the design phase reduces cost and effort, when compared to integrating security at the end i.e. at implementation phase. Designing for Security involves Architecture design. One good practice for Architectural design is: What is an accepted good practice when designing secure systems.

1.3 LAYERED SECURITY ARCHITECTURE

A Layer is an architectural pattern for security under category structure. Example security pattern is Secure Access Layer. A multi-layered Security Architecture will be helpful for modeling complex system. We require holistic system modeling to structure protection with multiple safeguards at many layers and locations to provide defense-in-depth. IT-T X.805 includes like: Application area of domain, Application, Data etc. Layered design helps for reusability of components, maximizes maintainability of code etc. For three-layered design architecture, the typical layers include: Presentation layer, Business Logic Layer and Data Layer. Software Architecture Decision framework has layers in architecture like: Conceptual architecture, Logical architecture and Execution architecture. All these three layers has both behavioral view and structural view. Conceptual architecture is abstract, Logical architecture is detailed and Execution architecture is about process view and deployment view.

1.4 MODEL DRIVEN ARCHITECTURE (MDA)

Model is a collection of UML (Unified Modeling Language) diagrams, which depicts the software architecture of the Software Application to be developed. Model Driven Architecture (MDA) basic premise to design the system software architecture in terms of models of the system. Traditionally most of the Software or Web application development was done using MDA, as most of the models are reusable for various domains, based on their layered nature.

1.5 AGILE METHODOLOGIES

Agile Software Development is based on Agile Manifesto, having 12 significant principles. Important principles for our consideration are: Simple Design: Pattern-based design to keep the design simple and well known. Test Driven development uses Architecture based testing. Agile Software Development (ASD) methods as per Agile Manifesto, supports continuous feedback and accommodate changes in software requirements throughout the software development life cycle, support close collaboration between customers and developers, and enable early and frequent delivery of software features required for a system. Agile design problems involve making architectural corrections and changes are difficult as per the architectural decisions. Agile methods are iterative approaches with focus on incremental specification, design and implementation. Agile methods are characterized by quickly producing correct functionality, which implies no or little documentation, little testing, and absence of verification and validation. There exist different agile approaches such as eXtreme Programming (XP) and Scrum. However, security is not the objective of agile methods; instead, the focus is on the correctness and time. Agile method approaches aim on producing individual components of functionality and then integrating them. However, even if individual components are secure, the aggregation of that component will not necessarily result in a measurably secure software system. Moreover, security expertise is absent, since security professionals are rarely involved in development teams.

Agile modeling involves emphasizes on people collaboration, both developers and customers are involved in Software development. Common features of Agile are:

Customer involvement, Incremental development and Delivery, Flexible Simple design and architecture, which improves maintainability.

1.6 DESIGNING SOLUTIONS

Software Engineering modeling methods can be depicted traditionally as a mathematical model. Software engineering problems requires solvations in terms of Solutions, especially in Design Phase.

1.7 DEPENDABILITY

Security engineering deals with building systems that stay dependable whether faced with error or malice. [10] Dependability generally means Risk Management, Trust negotiation, Privacy Management etc. In today's Organizational Information Systems Security in terms of dependability had taken an emergent new Dependable Security area called Privacy Management.

Privacy Management

Privacy of a User (Customer or Employee) of an Organization implies protection of their Personal Information or their Personal requirements from disclosure to un-authorized or un-intended entities. Privacy problems are inevitable during System Development Life Cycle, especially at Testing Phase. Data Privacy plan is required for Test Data Management, with Data Privacy Requirements, and they need to be defined and documented.

II. PAGE LAYOUT(SIZE 10 & BOLD)

2 EVOLUTION OF SOFTWARE ENGINEERING AS WEB ENGINEERING

Traditional Software Engineering Principles had quickly became inadequate to address the explosion of Web Applications, Web enabled applications and Web based applications etc. in the early 1990's. Hence to address specific problems raised by Web applications, suitable engineering principles available at that time, were tailored for web applications and those engineering principles that are validated to be effective are called as Web Engineering.

3 NEXT GENERATION SECURE WEB ENGINEERING APPLICATION (NGSWEA) – CASE STUDY

Software Engineering research validations using case studies approach has been proved to be more effective [7].Next Generation Web Engineering Applications uses integrated technologies like Agile Modeling, Web Services and Web 2.0. It provides architecture of providing security to NGSWEA in terms of Secure Agile Modeling, Secure Web Services Design, and Secure Web 2.0 Services Authorization, authentication, Privacy Security Requirements Management.

3.1 AGILE MODELING

As all the Web Applications developed using Web Engineering Principles are significantly using Agile Modelling Principles, Agile Security Design Model is required for Secure Web Engineering importantly for Web 2.0 Services.

3.2 WEB SERVICES

Web Services is the way next generation web applications are built. In a nutshell, on the web, software applications are going to provide services. As an example, If a user goes abroad for attending an International Conference, if he makes flight reservation using an Web application (web service), then associated web applications or web services for this user, based on his flight reservation, like hotel booking, cab booking are linked with flight reservation. The significant advantage of web services is that it works with and completely supports XML (eXtensible Mark up Language) based on transmission of xml data on existing web protocols like hyper text transfer protocol (http), data access is done using Simple Object Access Protocol (SOAP), registry uses Universal description discovery and integration (UDDI) protocol.

Figure 1.3 provides Web Services Application Architecture. It has major components like Client Application, Service Provider, and Service Registry. Client application finds and chooses a service available from Service Registry. Server is a service provider which publishes their services to the Service Registry. Server implements the web services and based on the request of client, it provides response to client.

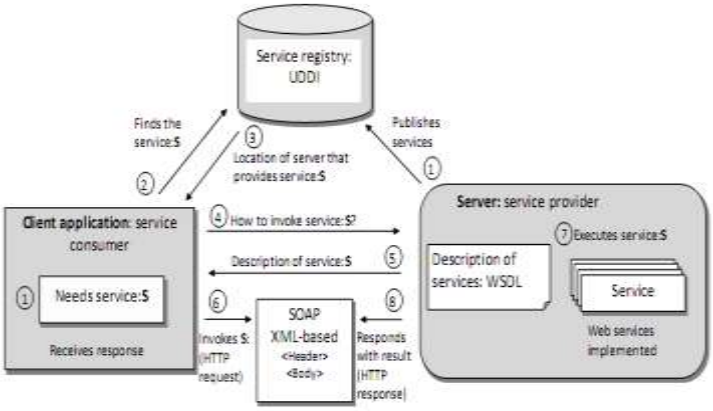

Fig 2: Web Services Architecture

Figure 1.4 provides Use Case model for Web Service, which depicts the design model of earlier Web Services Architecture. The actors involved in web services operation are Service Requestor (Client), Service Registry, Service Provider (Server). The use cases which provide web services functionality are Query Services, Discover Services, Bind Services, Publishes Services in Registry, Unpublished services in registry.

Fig 3 Use Case Model for Web Service Operation

Figure 4 provides Sequence diagram for Web Services operations. The sequence of steps involved in web services operations are: the actors are service requestor (client), service registry and service provider (server). Service provider publishes their offered services to service registry. Service requestor queries his required web service to service registry. Service Registry replies a service response to Service Requestor. Service Requestor discovers or chooses from list of available services to service registry. Service registry replies service description to service requestor. Now service requestor is binded (client) to service provider (server). Server sends a response to client. Eventually service can notify un publish service to Service Registry.

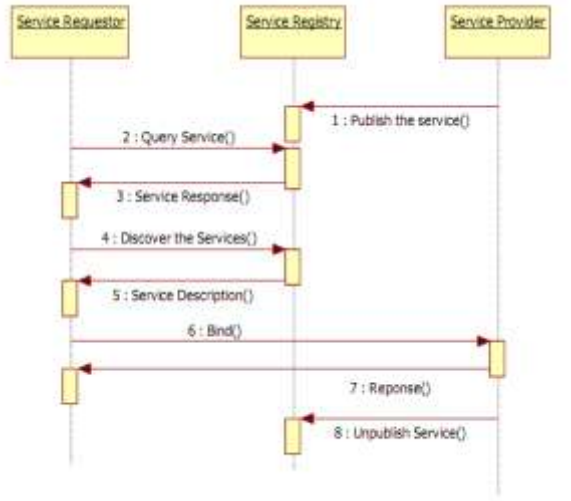

Fig 4 Sequence diagram for Web Services Operation

3.3 WEB 2.0

Web 2.0 has made web as a platform and it uses AJAX (Asynchronous Java Script and XML). Popular Web 2.0 applications include Spatial Google Maps.

Privacy for Web 2.0: Web 2.0 Privacy dimensions are Information Collection, Information usage, Information storage, Information disclosure, Information Security, Information Access Control, Information Monitoring, and Policy Changes. Each of these dimensions has specific Requirements/ assurance/guarantee.

Table 1.1 provides Architectural Styles as applied to NGSWEA case study. Layered Architecture Style is used for NGSWEA (Agile modeling, Web Services, Web 2.0) for structure. Service Oriented Architecture implemented as a Web Service is the architectural style used for communication.

Table 1 ARCHITECTURAL STYLES

Pattern Category	Examples of Architectural Styles
Structure	Layered Architecture
Communication	Service Oriented Architecture using Web services

Table 1 provides Security Pattern Classifications for Application Architectures for NGSWEA. Various Job Positions considered for Security Engineering here are Designer, Architect, Developer. Patterns used for designer are Password, Authenticator and Authorizer. Patterns used for architect are policy design and implementation. Patterns used for developer are Secure Data Buffer**.**

NGSWEA Applications user Web 2.0 services using Agile Modeling have rich user experience, CML Web Services applications, uses XHTML (eXtensible Hypertext Markup Language), CSS (Cascading Style Sheets) and AJAX (Asynchronous Java Script and XML) for user presentation interface.

4 RESEARCH MOTIVATION AND PROBLEM STATEMENT AND BACKGROUND TO THE PAPER OBJECTIVE AND SCOPE OF THE PRESENT STUDY

Traditionally, Software Engineering Problems were treated by both theoretical and empirical methodologies. The former is characterized by abstract, inductive, mathematics-based, and formal-inference-centered studies; while the latter is characterized by concrete, deductive, data-based, and experimental-validation-centered studies. This research involves in theoretical designing of Secure UML models using Agile Modeling. Also the results will be validated with experimental work on Next Generation Secure Web Engineering Application using Web 2.0 Services Security Architectures for Dependable Privacy Requirements on a Secure Stock Market Web application case study.

Problem Definition

This research objective is to motivate analysis and design addressing the novel idea and innovative implementations of Security Engineering for Software (Web) Engineering using proposed Agile Security Model which is layered design for Dependable Security Privacy Requirements, with a validation on an exemplar case study of NGSWEA of a Secure Stock Market application.

5 ORGANIZATION OF THE RESEARCH WORK

The research has been done at various stages and paper is organized into seven papers.

1. Introduction to various topics of paper like software architecture, security architecture, layered pattern, model driven architecture, agile methodologies, designing solutions, dependability, privacy

management, web engineering, Web Services and Web 2.0 are discussed. This leads to design of problem statement.

2. a detailed literature survey was conducted on the important topics of paper to find out basis for the paper.
3. discussion of theoretical analysis of designing of MDA extended Agile Security Architectures using layered design is provided. Initial case study validations on simple secure Web Services design using proposed agile security model using pair programming and test driven development.
4. Design and Implementation of our proposed model for Agile Modeled Layered Security Architectures for Security Requirements, along with Agile security patterns. We had validated on case study Next Generation Secure Web Engineering Applications, using Agile Modeling for Web Services, Web 2.0 Services Authentication using Secure Socket Layer (SSL) etc.
5. Dependability (Privacy Requirements) for Agile Modeled Layered Security Architectures is discussed. This is validated on case study of Web 2.0 Services Privacy Management. Finally validation for Secure Web Engineering using Agile Modeled Layered Security Architecture Solutions is discussed.
6. a detailed case study for Next Generation Secure Web Engineering Application, Secure Stock Market Web Engineering Application, using Agile Modeled Layered Security Architecture for Dependable Privacy Security Requirements.
7. the summary, conclusion and future work of the paper is presented.

6 SUMMARY AND CONCLUSION

The theme of the paper 1 is to recollect about basics of the paper with some main concepts like Software Engineering, Security Engineering, Software Architecture, Security Architecture, Layered Pattern, Model Driven Architecture (MDA), Agile Software Development, Designing Solutions, Dependable Privacy Requirements, Next Generation Secure Web Engineering Applications Case Study using Agile Modeling, Web Services, and Web 2.0. Also, motivation of the paper, organization of the paper is provided in this paper. In the next paper a detailed literature survey study is given about these concepts.

REFERENCES

1. Grady Booch, James Rumbaugh, Ivar Jacobson," The Unified Modeling Language User Guide", Pearson education, ISBN 81-7808-169-5, 1999, PP. 167.
2. Remco C de Boer, Hans van Vliet," On the Similarity between Requirements and Architecture", the Journal of Systems and Software, August 2008, PP. 1-6.
3. Erich Gamma, "Design Patterns Elements of Reusable Object Oriented Software", Addison Wesley Publishers 2009, PP. 102.
4. Marwan Abi-Antoun, Jeffrey M.Barnes,"Enforcing Conformance between Security Architecture and Implementation", CMU-ISR-09-113, April 2009, PP. 1-16.
5. Pekka Abrahamsson, Michele Marchesi, Frank Maurer," Agile Processes in Software Engineering and Extreme Programming", 10 th International Conference XP 2009, Springer Proceedings book Italy May 25-29 2009, PP. 2-3.
6. Patrizio pellicccione, Paolo Inverardi, Henry Muccini, "CHARMY: A Framework for design and verifying architectural specifications", IEEE Transactions on Software Engineering, Vol. 36, No. 3, May/June 2009, PP. 325 – 346.
7. Sabine Buckl, Ulrik Franke, Oliver Holschke, Florian Mattjes, Christian M.Schweda, Teodor Sommestad,"A Pattern-based Approach to Quantitative Enterprise Architecture Analysis", Proceedings of 15 th American Conference on Information Systems, San Francisco, California, August 6 – 9 2009 PP. 1-11.
8. Amund Hunstad, Jonas Hallberg, "Design for securability – Applying Engineering Principles to the design of security architectures", 2002, PP. 1-6.
9. Muchai Mutlugam, Ibrahim Sogukpinar, "Security Architecture for web based Health Insurance Systems", IEEE International Conference, 2006, PP. 24.
10. Simon G.Brown, Frederick Yip, "Integrating Pattern Concepts & Network Security Architecture" IEEE 2006, PP. 1-8.
11. Durai Pandian M et. al, "Information Security Architectures – Context aware Access Control Model for Educational Applications", International Journal of Computer Science and Network Security, December 2006, PP. 112 - 118.
12. Gunnar Peterson, "Security Architecture Blueprint", Arctec Group LLC, 2007, PP 1-12.
13. E.B. Fernandez, M.M. Larrondo, M. Vanhilst, "A Methodology to Develop Secure Systems Using Patterns", Idea Group Inc. 2007, PP. 107-126.
14. Nobukazu Yoshioka, Hironori Washizaki and Katsuhisa Marutama, "A Survey on security patterns", National Institute of informatics special issue No.5 2008, PP. 35-47.
15. Spyros T. Halkidis, Nikolaos Tsantalis, Alexander Chatzigeorgiou, George Stephanides, "Architectural Risk Analysis of Software Systems Based on Security Patterns", 2008, IEEE Transactions on dependable and secure computing, vol. 5, no. 3, PP. 129-142.
16. S. Aravind Krishna, "Secure Application Architecture for Mobile Police Force", CSI Communications May, 2009, PP. 56-58.
17. Stephan Bode, Anja Fischer, Winfried Kuhnhauser, Matthais Riebisch,"Software Architectural Design meets Security Engineering", IEEE 2009 16 th Annual IEEE Conference and Workshop on the Engineering of Computer Based Systems, PP. 109 – 118.
18. S.Mischelle Oda, Huirong Fu, Ye Zhu, "Enterprise Information Security Architecture A Review of Frameworks, Methodology, and Case Studies", IEEE 2009 (978-1-4244-4520-2/09), PP. 333-337.
19. A.Cenys, A.Mormantas, L.Radvilavicius, "Designing role-based access control policies with UML", Journal of Engineering Science and Technology Review 2009 PP. 48-50.
20. Michael VanHilst, Eduuardo B.Fernandez," A Multi-Dimensional Classification for Users of Security Patterns", Journal and Research and Practice in Information Technology, Vol.41, No.2, May 2009, PP. 87-97.