

# Designing Dependable Agile Layered Security Architecture Solutions – Web 2.0 Services Case Study

Dr.D.Sravan Kumar<sup>1</sup> M.Upendra Kumar<sup>2</sup> and D.Shravani<sup>3</sup>

<sup>1</sup>Principal and Professor CSE KITE Women's College of Professional Engineering Sciences Hyderabad A.P. India  
Email: [dasojusravan@yahoo.co.in](mailto:dasojusravan@yahoo.co.in)

<sup>2</sup>Research Scholar JNTUH and Associate Professor CSE MGIT Hyderabad A.P. India  
Email: [uppi\\_shravani@rediffmail.com](mailto:uppi_shravani@rediffmail.com)

<sup>3</sup>Research Scholar R.U. Kurnool and Assistant Professor CSE MIPGS Hyderabad A.P. India  
Email: [sravani.mummadi@yahoo.co.in](mailto:sravani.mummadi@yahoo.co.in)

**Abstract**—Software Security Engineering is about building systems to remain dependable in the face of malice, error, or mischance. To develop secure software systems, security should not be considered as a pure technical issue which is added as an afterthought, but security considerations must be integrated into the software engineering practice. Most attacks to software systems are based on vulnerabilities caused by poorly designed and developed software. The enforcement of Security at the Design phase can reduce the cost and effort associated with the introduction of security during implementation. There's no substitute for working software security as deeply into the development process as possible. Security Architecture is the knowledge in technologies needed to build secure platforms. Layered Security Architectures motivates various layers for dependability like Application domain, Application, Temporal, Distribution, Data and Resource. Agile Software development methods can integrate security into information systems by refining the security requirements. In this paper, we discuss about Agile Design principles for enhancing security of Layered Security Architecture solutions. We validate this through exemplar Web 2.0 Services Security Architectures for authentication over SSL/TLS.

**Index Terms**—Layered Security Architectures, Agile Software Development, Designing Dependable Architectures, Web 2.0 Services Security

## I. INTRODUCTION TO SECURITY ARCHITECTURES

Architecture, whether system or application, are composed of abstractions (interfaces) and their implementations. Security Architectures are architectures which enable implementations that are resilient to an appropriate and broad-based spectrum of threats [1]. An evaluation of Security Architectures requires understanding these threats; the tradeoffs between different system goals, including between security and non-security goals; the long-term appropriateness of its interfaces; and the implementations it allows. The best interfaces are those that capture the most important issues, enable different implementations, and are flexible enough to adapt (are be adapted) to different threats. Two well-known issues are particularly important: First, complexity is a source of security holes. Second, security

is a matter of the weakest link. Because of the need to balance complexity versus protections, these tradeoffs are often controversial. Other tradeoffs include performance, usability, and flexibility. The design and evaluation of Security Architectures is of fundamental importance to security and yet many of our fundamental architectures were created when security was less appreciated and less well understood. Since it is notoriously difficult to add security after the fact, our systems are far too susceptible to attack. Moreover, architectures, because they are broad based, are difficult to understand [2].

### A. Layered Security Architectures

Various layers of a Security Architecture (like Telecommunication Standardization Sector-International Telecommunication Union ITU-T X.805) include [3]: Application Domain, Application, Temporal, Distribution, Data and Resource. (Refer Table1)

TABLE 1. LAYERED SECURITY ARCHITECTURES

Application Domain
Application
Temporal
Distribution
Data
Resource

The temporal layer will address time-based security and will feature workflow related solutions. The distribution layer will address communication-based security. The Open Systems Interconnect (OSI) will best fit into the distribution layer. The data layer addresses secure data storage. An obvious technology for the data layer would be database management systems (DBMS). The resource layer is used to include factors that are instrumental in the functioning of a system, but which operate independently from the rest of the system. The application layer will provide for an external view of the system. It will address user and user group roles, policies as well as any inter-organizational issues. The application domain refers to the context in which a system operates.

An example of an application domain would be the Internet.

A Layer can also be considered as a security architectural pattern of the category "Structure". For example "Secure Access Layer" builds our application security around the existing Operating System, networking, and database security mechanisms of the computer [4]. If these security infrastructures do not exist, then should build our own lower-level security mechanism. On top of the lower-level security foundation, we should build a secure access layer for communicating in and out of the program. The important point about this pattern is to build a layer to isolate the developer from change. This layer may have different protocols depending on the types of communication that need to be done. For example, this layer might have a protocol for accessing data in an Oracle database and another protocol for communicating securely with an Apache server through the Secure Socket Layer (SSL). The basic theme of this pattern is to componentize each of these external protocols so they can be more easily secured. Defense in Depth can also be implemented as a layered Defense, where layers include: Router, Firewall, Intrusion Detection System (IDS), Virus, Hosts, Policies and Users.

## II. AGILE SECURE SOFTWARE DEVELOPMENT

An Agile approach embraces change as part of the software development process [5]. In most approaches, change is usually considered a bad word. Agile developers work in pairs, create many prototypes of their solutions, and incorporate user's feedback throughout the entire process. Agile software development has encouraged developers to tailor their methods to meet their specific needs. Agile Modeling using Unified Modeling Language is geared towards small development projects with tight deadlines, like building Web front ends. Agile implies that the security requirements have changed because the threats have changed. Defensive security uses OODA cycle [Observe (Senses change in its settings), Orient (analyze meaning and importance of this change), Decide (determine an ideal strategy taking advantage of change), Act (implementing this strategy)]. Key security elements in agile software development for Requirements Analysis phase include: Security-relevant subjects, Security-relevant objects, Security classifications and Subjects and Risk Management.

## III. AGILE DESIGN PRINCIPLES

Design phase should ensure that proper security requirements are included in the design phase: Designers include security subjects and security objects using the notation of agile design diagrams. They incorporate the security classification of security subjects and objects according to the modeling notation. They apply risk management as necessary to prioritize the features. In eXtreme Programming (XP), the initial design is defined with the System Metaphor [6]. The System Metaphor is a short description about how the system works. It is

written in a simple natural language that facilitates communication among the stakeholders of the development. The design is further implemented with the use of models like user stories, acceptance tests and CRC cards. These models are refined in each iteration i.e. as the work is incremental; the additions of new system components or user's requirements are also modeled iteratively. An important design principle is the Principal of Least Privilege. It states that a subject should be given only those privileges that it needs in order to complete its task. There are tasks that need to be performed only at an elevated privilege. Any computer program must always remain in a least privileged state. When there is a need, it will elevate the privilege only for the duration needed. Once the privileged function is complete, it must return to the state of least privilege.

## IV. DEPENDABLE SOLUTIONS

Key analysis and design consideration for security of architectures deals with, "How well the system authenticates the users and protects the application and data elements?" Various Software Engineering paradigms exist such as model-driven, aspect-oriented and agent-oriented. New methods and techniques that are required should support the formal (and simultaneous) modeling, reasoning and analysis of security and functional requirements, and transformation of such (security and functional) requirements to a design that will satisfy them along with support for traceability and validation of the proposed solution.

Tool support: A tool should not only support developers in modeling and reasoning about security (and functional requirements) during the analysis stage, but it should help to transform the requirements to design, check the consistency of the proposed solution and to validate the security functionalities of the proposed solution and to validate the security functionalities of the proposed solution and also validate the security functionalities of the proposed solution against the security solutions of the system.

## V. WEB 2.0 SERVICES CASE STUDY

### A. *Web 2.0 Services Security Architectures*

According to WWW Consortium a web service is defined as, "A Web Service is a software application identified by a URI (Uniform Resource Identifier), whose interface and bindings are capable of being identified, described and discovered by XML artifacts and supports direct interactions with other software applications using XML based messages via Internet-based protocols". [7] Web Services Security Architectures have three layers viz. Web Service Layer, Web Services Framework Layer (.NET or J2EE), Web Server Layer. Web 2.0 increases web based access to data processing particularly on the client side (AJAX Asynchronous JavaScript and XML) that enables web applications which contain enriched functionality. Web 2.0 technologies have wide range of

technologies and protocols which enable Web architectures to have greater access to data and functions.

**B. Implementation and Validation**

In this case study we take a look at WS-security(Web Services Security) agreements and use the Microsoft implementation, known as WSE 3.0(Web Services Enhancements), in conjunction with visual studio 2005 to produce a web service that requires authentication. There are a number of ways to secure a web service. A simple way might be to use Secure Socket Layer (SSL) together with client side certificates to ensure that the potential user is someone allowed to call the services method. The problem with this approach would be to pass the user credentials as part of the method call. Here we implemented web 2.0 services security using visual studio 2005 with modules: Adding Policy, Adding the Custom Authentication (using traditional X.509 certificates and XML counterpart), creating a Client. SSL provides a secure tunnel through which Simple Object Access Protocol (SOAP) message will pass while WS-Security provides message protection by extending SOAP. The main advantages of this are simplicity of software design and code, correctness, while providing security against sniffing and confidentiality attacks. We validated the results using correctness testing. The components used at message layer [8] are shown in Table 2. The authentication mechanism is shown in Figure 1. Class diagram for SSL/TLS security is shown in Figure 2.

TABLE 2. COMPONENTS AT MESSAGE LAYER

Ws-security
SOAP
SSL/TLS

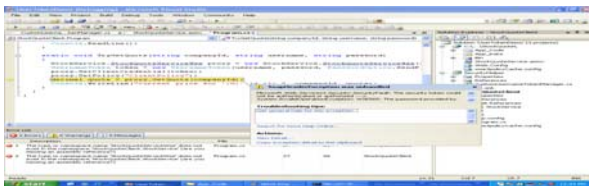


Figure 1. Error display when user enters wrong username / password

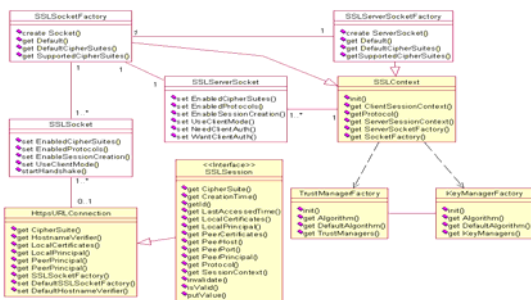


Figure 2. Class diagram for SSL/TLS layer implementation

We implemented four classes: Factory Classes (SSLSocketFactory, SSLServerSocketFactory) and Socket Classes (SSLSocket and SSLServerSocket). Here SSL protocol enables Client side to send back certificates chain and then a signature. Client side authentication also is same but we require making additional configuration to our defined classes.

For details of implementation code, documentation and detailed UML diagrams, please refer to the website <http://sites.google.com/site/upendramgitcse/>

CONCLUSIONS

In this paper, we urged the need for integrating security and software engineering solutions using agile layered security architecture dependable secure design methodology. Further work includes addressing research questions like: “How does a failure of a specific security service at a specific layer impact other (interdependent) layers? Also how does a successful implementation of a security service affect rest of the system? How can agile methods be used to generate effective security requirements? In what ways can these agile methods change the development of security requirements? How is the outcome of emergent security development different from more traditional forms?”

ACKNOWLEDGMENT

The authors wish to thank the following students of Computer Science Engineering department at M.G.I.T. for implementing these concepts: Ch.Venkatashilash, S.Vamshidher Reddy and G.Sri Harsha.

REFERENCES

- [1] Gunnar Peterson, “Security Architecture Blueprint”, Arctec Group LLC, 2007.
- [2] Spyros T. Halkidis, Nikolaos Tsantalis, Alexander Chatizigeorgiou and George Stephanides, “Architectural Risk Analysis of Software Systems Based on Security Patterns,” *IEEE Transactions on Dependable and Secure Computing*, vol. 5 no. 3, pp. 129–142, July-September 2008.
- [3] Heiko Tillwink and Martin S Olivier, “A Layered Security Architecture: Design Issues”, in Proceedings of the Fourth Annual Information Security South Africa Conference (ISSA2004), July 2004.
- [4] Mouratidis and Giorgini, *Integrating Security and Software Engineering: Advances and Future Vision*. Idea Group Publishing Inc., 2007.
- [5] Russo, Scoto, Silliti and Succi, *Agile Technologies in Open Source Development*. IGI Global, 2010.
- [6] Asoke K. Talukder and Manish Chaitanya, *Architecting Secure Software System*. CRC Press, 2009.
- [7] Massimo Bartoletti, Pierpaolo Degano, Gian Luigi Ferrari and Roberto Zunino, “Semantics Based Design for Secure Web Services,” *IEEE Transactions on Software Engineering*, vol. 34 no. 1, pp. 33–49, January-February 2008.
- [8] Anoop Singhal and Theodore Winograd, *Guide to Secure Web Services*. NIST Draft (800-95), September 2006.
- [9] Rolf Oppliger, Ralf Hauser and David Basin, “SSL/TLS Session-Aware User Authentication,” *IEEE Computer*, pp. 59–65, March 2008.