ISSN:0975-9646

Security Patterns for Web Services Mining Agile Architectures

A.V.Krishna Prasad[1] Dr.S.Ramakrishna[2] D.Shravani[3]
[1]*Associate Professor Department of Computer Science MIPGS Hyderabad, A.P. India*
Email: kpvambati@gmail.com
[2] *Professor Department of Computer Science S.V.University Tirupathi A.P. India*
Email: drsramakrishna@yahoo.com
[3]Research Scholar R.U. Kurnool and Assistant Professor CS MIPGS, Hyderabad A.P. India
Email: sravani.mummadi@yahoo.co.in

***Abstract*— The importance of the software security has been profound, since most attacks to software systems are based on vulnerabilities caused by poorly designed and developed software. Design flaws account for fifty percent of security problems and risk analysis plays essential role in solid security problems. Security Patterns are proposed that offer the security at the architecture level in analogy to design patterns. Determination of up to what extent specific security patterns shield from known STRIDE attacks to architecture is a major task. In this paper, we want to validate security patterns approach for architectures, using Executable UML Model-driven Architecture development and Layered Security Architecture. These results encourage the new research area of Web engineering Navigational Development Technique. Initially we look at authorization using MDA Executable UMLSec. Finally, we validate this by implementing security patterns using Agile Modeling.**

***Index Terms*—Layered Security Architectures, Agile Modeling, Security Patterns, Model Driven Architecture, Executable UML.**

I. INTRODUCTION TO SECURITY PATTERNS FOR SECURITY ARCHITECTURES

Security has become an important topic for many software systems. Security Patterns are reusable solutions to security problems. [1] Although many security patterns and techniques for using them have been proposed, it is still difficult to adapt security patterns to each phase of software development, and integrating them into Software Architecture design for modeling attack design. Various Patterns for Security Architectures includes: Single Access Point, Check Point, Roles, Session, Full View with Errors, Limited View, Secure Access Layer, Least Privilege, Journaling and Exit Gracefully. The importance of software security has been profound, since most attacks to software systems are based on vulnerabilities caused by poorly designed and developed software. Furthermore, the enforcement of security in software systems at the design phase can reduce the high cost and effort associated with the introduction of security during implementation. For this purpose, security patterns that offer security at the architectural level have been proposed in analogy to the well-known design patterns. We need to perform risk analysis of software systems based on the security patterns that they contain. The first step is to determine to what extent specific security patterns shield from known attacks at the design phase. Successful software and software systems are directly attributable to elegant and efficient modeling and design. [2] Models let users, architects and developers create readily understandable representations of complex object-oriented systems, before development begins. Sometimes these representations are visual (for example, class diagrams) and sometimes they are non-visual (for instances, use cases). A good analysis model for a portion of a complex can be abstracted and become an analysis pattern that can be used in other applications. Their use can save time and improve the quality of a system. An important advantage of analysis phase Semantic Analysis pattern is that they can be combined easily with security patterns, resulting in authorized applications. The security defined in the conceptual model can be enforced in the design model using security patterns at the lower architectural levels, including security patterns, components, distribution, and database adapters. We are currently developing more security patterns, including patterns for secure brokers and for Web Services as well as collection of patterns. The combination of multilayer architectures with patterns provides a framework to develop a systematic and reusable approach to building systems that satisfy specific non-functional requirements. Security patterns embody good design principles and by using them, the designer is implicitly applying these principles. Work is needed to add more patterns in each level and to collect and unify these patterns. We also need to define guidelines to apply the methodology in a real environment; for now, we are applying it to specific examples, such as distributed medical records, Internet voting, and distributed financial institutions. Refer to Table 1 which consists of Architectural Styles and Refer to Table 2 for Security Pattern classification for Application Architectures.

TABLE 1. ARCHITECTURAL STYLES

Category	**Architectural Styles**
Communication	SOA(Service Oriented Architectures) , Message Bus
Deployment	Client/Server, N-Tier, 3-Tier
Domain	Domain Driven Design
Structure	Component-Based, Object-Oriented, Layered Architecture

TABLE 2. SECURITY PATTERN CLASSIFICATION FOR APPLICATION ARCHITECTURES

Stakeholder	Function	Data	Test
Architect	Checkpointed System Policy Enforcement Point Replicated System Secure Preforking Single Access Point		
Designer	Authenticator Password Synchronizer	Exception Shielding Subject Descriptor	
Developer		Safe Data Buffer	Grey Hats

Secure Software Architecture using MDA: A basic premise of Model-Driven Development (MDD) is to capture all important design information in a set of formal or semiformal models, which are then automatically kept consistent by tools. [3] Research community found the MDD approach to be deficient in terms of modeling the architectural design rules. It does not offer a satisfactory solution as to how architectural design rules should be modeled. As a result developers have to rely on time-consuming and error-prone manual practices to keep a system consistent with its architecture. To realize the full benefits of MDD, find ways of formalizing architectural design rules, which then allow automatic enforcement of the architecture on the system model. There exist several approaches to MDD, such as OMG's MDA, Domain Specific Modeling (DSM), and Software factories from Microsoft. MDA (Model-Driven Architecture) prescribes that three models or sets of models shall be developed as: The Computationally Independent Model(s) (CIM) captures the requirements of the system The Platform-Independent Model(s) (PIM) captures the systems functionality without considering any particular execution platform; The Platform-Specific Model(s) (PSM) combines the specifications in the PIM with the details that specify how the system uses a particular type of platform. The PSM is a transformation of the PIM using a mapping either on the type level or at the instance level. A type-level mapping maps types of the PIM language to types of the PSM language. An instance-level mapping uses marks that represent concepts in the PSM. When a PIM shall be deployed on a certain platform, the marks are applied to the elements of the PIM before the transformation; MDA does not directly address architectural design or how to represent the architecture, but the architecture has to be captured in the PIM or in the mapping since the CIM captures the requirements and the PSM is generated from the PIM using the mapping.

Web engineering is a new research line in software engineering that covers definition of processes, techniques, and models suitable for web environments in order to guarantee the quality of results. The research community assumed the Model-Driven paradigm to support and solve some classic problems detected in web developments but there is a lack in web requirements treatment. Therefore, NDT (Navigational Development Technique) was developed which deals with requirements in web systems. It is based on conclusions obtained in several comparative studies and it tries to fill some gaps detected by research community. It analysis how web engineering can be applied in the enterprise environment. The approach offers a web requirements solution based on a Model-Driven paradigm that follows the most accepted tendencies by web engineering. Here, security patterns are proposed that offer the security at the architectural level in analogy to design patterns. The first step is to determine to what extent specific security patterns shield from known attacks. This information is fed to a mathematical model based on the fuzzy-set theory and fuzzy fault trees in order to compute the risk for each category of attacks. Estimates are proposed for the resistance of the examined security patterns to Spoofing, Tampering-with-data, Repudiation, Information-disclosure, Denial-of-service, and Elevation-of-privilege attacks. We had proposed a methodology for quantifying the security level of a software system based on the implemented/missing security patterns. [4] Moreover, the estimation can be performed already at the design phase. Thus, design problems can be detected at an early stage, which reduces the cost compared to the introduction of security during implementation. The comparison of two e-commerce systems having the same functionality, one without and one with security patterns, has shown that the nonsecure application has a high risk of being affected by each category of STRIDE attacks, where as the secure application has a significantly lower risk.

II. LAYERED SECURITY ARCHITECTURE PATTERNS

The importance of the software security has been profound, since most attacks to software systems are based on vulnerabilities caused by poorly designed and developed software. [5] Design flaws account for fifty percent of security problems and risk analysis plays essential role in solid security problems. To improve the quality of software systems, design patterns are important in object-oriented programming because they offer design motifs, elegant solution to recurrent design problems. DeMIMA (Design Motif Identification Multilayered Approach), an approach to semi automatically identify micro architectures that are similar to design motifs in source code and to ensure the traceability of these micro architectures between implementation and design. DeMIMA consists of three layers: two layers to recover an abstract model of source code, including binary class relationships, and third layer to identify design pattern in the abstract model. Nuemann and Parker organized systems into eight layers for security analysis: External environment, user, application, middle, networking, operating system, hardware and internal environment. [6] Neumann’s model needs simplification to reason about systems, especially to construct an executable model. Adding sub-layers, this architectural model can be reduced to three layers Semantic, Logical and physical. Semantic layer at the top includes people and organizations along with their goals. Logical layer in the middle contains computers, networks and software. Physical layer at the bottom represents the physical existence that all entities have in the real world. Every layer has a different concept of location, representing the separate conceptual scope and connectivity of systems and entities at each layer.

III. AGILE SECURE PATTERNS

Because of several vulnerabilities in software products and high amount of damage caused by them, software developers are enforced to produce more secure systems. Software grows up through its life cycle, so software development methodologies should pay special attention to security aspects of the product. [7] Agile methodologies for security activities include applying agility measurement and applying an efficient agility reduction tolerance (ART). Using this approach method engineer of the project can enhance their agile software development process with security features to increase product's trustworthiness. A secure system is one that is protected against specific undesired outcomes. Delivering a secure system, and particularly, a secure web application, is not easy. Integrating general-purpose information systems development systems with security development activities could be a useful means to support these difficulties. Agile processes, such as Extreme programming, are of increasing interest in software development. Most significantly for web applications, agile processes encourage and embrace requirements change, which is a desirable characteristic for web application development. Agile methods include Feature Driven Development (FDD) and mature security methods, namely risk analysis, and integrate them to address the development of secure web applications. This approach key feature includes: a process capable of dealing with the key challenges of applications development like decreasing life-cycle times and frequently changing requirements and an iterative approach to risk analysis that integrates security design throughout the development process.

IV. SECURITY PATTERNS IMPLEMENTED USING AGILE MDA AND EXECUTABLE UML

MDA Security Implementations: Analysis at the level of runtime architecture matches the way expert's reason about security or privacy better than a purely code-based strategy. However, the architecture must still be correctly realized in the implementation. Security ensures that information is provided only to those users who are authorized to possess the information. [8-15] Security generally includes the following: *Identification*: This assumes that the system must check whether a user really is whom he or she claims to be. There are many techniques for identification and it is also called as authentication. The most widely used is "Username/Password" approach. More sophisticated techniques based on biometrical data are like retinal or fingerprint scan; *Authorization:* This means that the system should provide only the information that the user is authorized for, and prevent access to any other information. Authorization usually assumes defining "user access rights", which are settings that define to which operations, data, or features of the system the user does have access; *Encryption:* This transforms information so that unauthorized users (who intentionally or accidentally come into its possession) cannot recognize it. Refer to Figure 1 (Class Diagram): User enters username and password to access information. Authenticator checks the username and associated password to know whether the user is really he or she claims to be. Authenticator allows the user depending on the check result. Authorizer checks this user type (for example, administrator) and associated access rights. Authorizer restricts the user to access the information. The entered username and password by any user will be transformed in an encrypted format so that any other user who is correctly logged in cannot recognize it. Therefore security class provides a key and algorithm used to encrypt the data. Its implementation Sequence Diagram works as: User enters the username and password which are encrypted and transferred to authenticator to verify correctness. Then access rights for specified user are checked to allow for accessing of information.

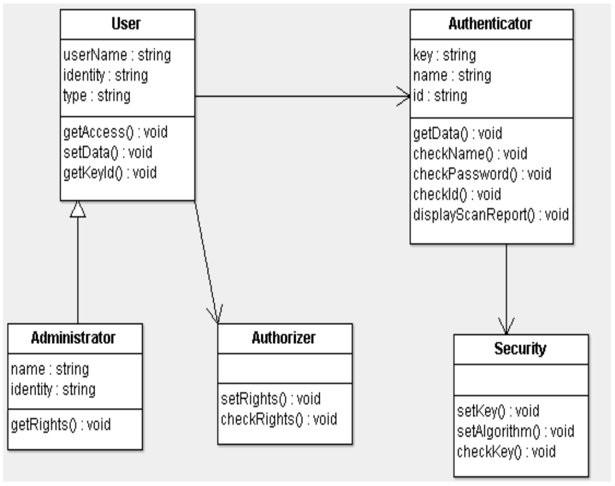

Figure 1. Class diagram for MDA authentication using Executable UML

V. IMPLEMENTATION AND VALIDATION

A. Agile Security Patterns Design

Agile Methodology with Security Activities:

There are some agile methods Extreme Program (XP), Scrum, TDD (Test Driven Development), FDD (Feature Driven Development). In order to restrain reduction of agility nature, a method has been introduced. The method consists of five steps as: Extracting Security Activities; Calculating Agility Degree of Security Activities; Integration of Agile and Security Activities; Activity Process Integration Algorithm; Agility Reduction Tolerance (ART)

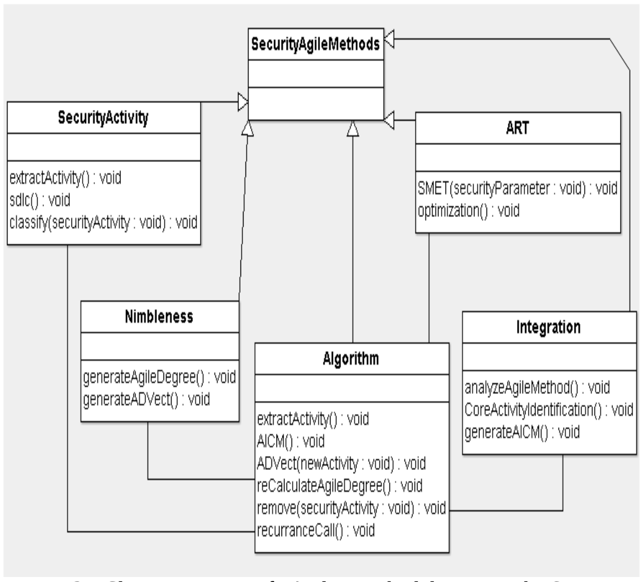
SecurityAgileMethods
SecurityActivity
extractActivity() : void
sdlc() : void
classify(securityActivity : void) : void
ART
SMET(securityParameter : void) : void
optimization() : void
Nimbleness
generateAgileDegree() : void
generateADVect() : void
Algorithm
extractActivity() : void
AICM() : void
ADVect(newActivity : void) : void
reCalculateAgileDegree() : void
remove(securityActivity : void) : void
recurranceCall() : void
Integration
analyzeAgileMethod() : void
CoreActivityIdentification() : void
generateAICM() : void

Figure 3. Class Diagram of Agile Methodologies with Security Activities.

First Security Activities are extracted from existing processes and guidelines from SecurityActivity class. The activities are named as "Security Activities" and these are used as basis for next steps. Classification of activities is done by understanding them in life cycle. Agility degree of activities is defined to measure their nimbleness. Agility degree for each activity is defined as its agile behavior. It represents level of activity's compatibility with agile methodologies. Grades between 0 and 5 are assigned in agility degree vector (ADVect).Then integration issues of agile and security activities are handled. By analyzing agile methodologies and identifying their core engine activities integration is done. Activity integration compatibility matrix (AICM) is generated with binary values. An algorithm to integrate security activities with organization's agile process is introduced in Algorithm class. Follows all steps activity by activity recursively. Finally agility reduction tolerance parameter and its optimization value are discussed in ART class. Tuning ART parameter is SMET's (Secure Method Engineer Team) art to keep a balance between security and weight of the software development process.

Agile Approaches to Design:

A modern concept of design modeling is performed through the design patterns. Design pattern is a solution to a problem of design that repeatedly occurs and that can be implemented in the code. The use of design practices and patterns is given as: Agile methodologies focus on incremental development without a single and large upfront design. BDUFA (Big Design Up Front Anti-pattern) is adopted. User changes idea on subject leading to gathering of new requirements and change in code and design. Agile design approaches can be readily identified to implement. Agile methods like Extreme Program, Scrum, TDD (Test Driven Development) and FDD (Feature Driven Development) are introduced in various cases or scenarios.

Agile Aspect Oriented Programming (AOP) Design:

AOP has a lot of similarity to agile software development methods: in particular, Extreme Programming and scrum. The basic idea of agile methods is to develop software iterations. Each of them resulting in software delivery. The main purposes of agile methods are adaptation to changing requirements to the product, and quick delivery, as opposed to using a traditional waterfall method .In a sense, from the viewpoint of a software process organization, using AOP can also be considered as one of the agile techniques. *Communication and respect:* AOP can be used with XP as follows. Each developer is responsible for some cross-cutting functionality to be implemented as an aspect. The team leader and the team discuss the AOP structure of the system and the manner of weaving the aspects to get an entire working system. System requirements can be distributed throughout the team as aspect stubs that at the initial stage or at any other moment, can be discussed, criticized by anybody, and updated upon mutual agreement. As for aspect implementation, according to the respect principle, nobody can modify as aspect developed by another programmer without prior discussion with the aspect developer. That will enable really safe and modular implementation of the collective code ownership principle. .If developer A remembers ,say that the developer B is the best expert on the security aspect, A will consult B before changing security functionality in B's aspect ,or,better,suggest the changes to B, and it will be B's final decision whether or not to include the changes. *Simplicity and courage:* Incremental development is quite suitable for applying AOP .The simplest version of the new feature implementation can be developed as an aspect .It will help us to correctly determine the join points at which to inject fragments of the new feature implementation .If there is a need which enhance the feature implementation, it is likely to be done only by modifying the aspect's actions only. The scheme of weaving the aspect will remain the same; it means that the XP team will not have to do big group changes in the entire code of product, and all changes will be localized in the aspect. *Feedback and test-driven development:* If a new functionality is designed, a new aspect should be developed. Prior to that, acceptance tests are developed for the functionality aspect. Implementation of the aspect is driven by its acceptance tests .Customers who are participating in the development process, according to XP principles ,will also think about system development and enhancement in terms of aspects.

Web Services Mining

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. Web Services Security Architectures have three layers viz. Web Service Layer, Web Services Framework Layer (.NET or J2EE), Web Server Layer.

For Web Servers security layer, important web servers such as the institution's main server and other publicly accessible web servers are major targets of attack from the Internet. Additional measures to protect these systems against malicious or accidental harm include making all updates on a staging server, running Common Gateway Interface (CGI) on a separate server, and having a hot backup server, and having a hot backup server. Using a staging server that is separate from the main server and make updates to the main server in a more controlled fashion. Placing CGI on a separate server prevents intruders from gaining access to the main server via insecure programs. A backup server that is regularly synchronized with the main server enables us to recover from an incident quickly by placing the primary server with the backup.

For applications and databases layer, it is not feasible to require through security design in all applications and databases deployed on a college or university network. Some applications and database guidelines are simple to require such as SSL/HTTPS for web-based services and logging facilities for auditing in the case of a problem. However, in many instances it is necessary to accept commercial products that have serious vulnerabilities (for example, MSSQL). Additionally, it may be too costly to implement security in some commercial systems (for example, Oracle) or in-house applications. Therefore, it is necessary to defend these systems at the network level. Using tiered database/application architecture and other middleware solutions when available can make it easier to defend these systems and protect the data they contain. More advanced requirements such as peer review of source code are difficult to enforce. Participating in the PKI Lite/Federal Bridge project may be feasible in some institutions, providing support for IPSec, S/MIME, and other security features such as access control. Other technologies such as VOIP and IP Security cameras can be difficult to secure and require special attention

Conclusions

In this paper, we discussed about Layered Security architecture Security patterns using MDA Executable UML, with a case study. An interesting extension to this work would be the automatic introduction of missing security patterns either at the design phase of a system being developed or in already implemented software systems.

For details of implementations source code (pseudo code) and documentation please refer to the web site http://sites.google.com/site/kpresearchgroup

Acknowledgment

The authors wish to thank the following for implementing these concepts: A.Madhuri, P.Radhika, P.Hari Krishna, and Ch.Venkat Abilash.

References

[1] Sabine Buckl, Ulric Franke, Oliver Holschke, Florian Matthes, Christian M.Schweda, Teodor Sommestad and Johan Ullberg, “ A Pattern-based approach to Quantitative Enterprise Analysis”, Proceedings of the Fifteen Americas Conference on Information Systems, San Francisco, California, pp. 1 – 11, August 6 – 9, 2009.

[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] Nobukazu Yoshioka, Hironori Washizaki and Katsuhisa Maruyama, “A Survey on Security Patterns”, Progress in Informatics, Special Issue: The Future of Software Engineering for Security and Privacy, No. 5, pp. 35 – 47, National Institute of Informatics, 2008.

[4] Michael Vanhilst, Eduardo B.Fernandez and Fabricio Braz,” A Multi-Dimensional Classification for Users of Security Patterns”, Journal of Research and Practice in Information Technology, Vol. 41, No. 2, May 2009, Australian Computer Society Inc.

[5] E.B.Fernandez, M.M.Larrondo-Petrie T.Sorgente, and M.Vanhilst,” A Methodology to develop Secure Systems Using Patterns”, Idea Group Inc. pp. 107 – 126, 2007.

[6] Munawar Hafiz, Paul Adamczyk and Ralph E. Johnson,” Organizing Security Patterns”, *IEEE Software*, July/August 2007, pp. 52 – 60.

[7] Anders Mattsson, Bjorn Lundell, Brian Lings and Brian Fitzgerald, “Linking Model-Driven Development and Software Architecture: A Case Study”, *IEEE Transactions on Software Engineering,* Vol. 35, No. 1, pp. 83 – 93, January/February 2009.

[8] Marria Jose Escalona and Gustavo Aragon, “NDT. A Model-Driven Approach for Web Requirements”, *IEEE Transactions on Software Engineering*, Vol. 34, No. 3, May/June 2008.

[9] Yann-Gael Gueheneuc and Giuliano Antoniol, “DEMIMA: A Multilayered Approach or Design Pattern Identification”, *IEEE Transactions on Software Engineering,* Vol. 34 No. 5, pp. 667 – 684, September/October 2008.

[10] Hossein Keramati, Seyed-Hassan and Mirian-Hosseinabadi, “Integrating Software Development Security Activities with Agile Methodologies”, *IEEE*, 99. 749 – 754., DOI 978-1-4244-1968-5/08. 2008

[11] Xiaocheng Ge, Richard F.Paige, Fiona A.C.Polack, Howard Chivers, Phillip J.Brooke, “Agile Development of Secure Web Applications”, ACM ICWE 06, pp. 305 – 312, July 11 – 14, 2006, Palo Alto, California, USA.

[12] Gustav Bostrom, Jaana Wayrynen, Marine Boden, Konstantin Beznosov and Phillippe Kruchten, “Extending XP practices to support Security Requirements Engineering”, ACM SESS 06, pp. 11 – 17, May 20 – 21, 2006, Shanghai, China.

[13] Tore Dyba and Torgeir Dinagsoyr, “Empirical Studies of Agile Software Development: A Systematic Review”, Elsevier Science Direct Information and Software Technology, 2008.

[14] A.Cenys, A.Normantas and L.Radvillavicius,” Role-based Access Control Policies with UML”, Journal of Engineering Science and Technology Review, pp. 48-50, Kavala Institute of Technology, 2009.

[15] Marwan Abi-Antoun and Jeffrey M.Barnes, “Enforcing Conformance between Security and Implementation”, CMU-ISR-09-113, April 2009.