A Framework for Composable Security Definition, Assurance, and Enforcement

J. A. Pavlich-Mariscal
Advisors: S. A. Demurjian and L. D. Michel

Department of Computer Science & Engineering
The University of Connecticut, Unit-2155
371 Fairfield Road
Storrs, CT 06269- 2155
jaime.pavlich@uconn.edu, {steve,ldm}@engr.uconn.edu

Abstract. The objective of this research is to develop techniques that integrate alternative security concerns (e.g., mandatory access control, delegation, authentication, etc.) into the software process. The resulting model-driven framework preserves separation of security concerns from modeling through implementation, and allows security personnel to pick and choose security concerns to
concerns promotes security assurance, and should result in a reduction of the security defects in the final system. To achieve separation of concerns at the modeling level, concern-specific languages are defined to capture alternative security concerns. At the implementation level, aspect-oriented programming is used to integrate security concerns into an application's code, while preserving modularity. This composition seamlessly combines the chosen security concerns to realize an application's security infrastructure.

1 Introduction

In today's world, information and its secure access, as managed by software applications, is a critical asset of organizations. Within the software process, although some security requirements are addressed at early stages in development, most are discovered after functional requirements are defined and implemented[3]. Security concerns added at later stages in the software process, particularly post-implementation, can increase security defects in an application. For that reason, realizing security requirements at the earliest stages of the software process is crucial to deliver software applications whose information is secure, uncorrupted, and available. A useful technique to achieve this goal is *separation of concerns*, to distinguish all important concerns of an application into modular units that can be developed independently. In order to effectively separate concerns, software formalisms must provide *decomposition mechanisms* that can partition the software into simpler pieces that are easier to manage, and *composition mechanisms* to join all components together into a final complete system [13]. The premise is that isolating security specifications from the rest of the application, and providing adequate composition and decomposition mechanisms for them, can effectively integrate security into the software process.

To treat security as a separate concern, several key issues must be addressed:

Security modeling: Widespread software modeling languages (e.g., the unified modeling language, UML), do not specifically target security requirements. As a result, security is often tangled and spread throughout the design/code. Modeling languages like UML must be augmented in order to better conceptualize security and its relation to the overall design.

Security enforcement: As an application's complexity increases, so too do security errors and flaws, particularly for poorly modularized security concerns. Security requirements, modeled in a separate concern, must be transitioned to a well-structured enforcement mechanism via programming language support and techniques that modularize security at the code-level.

Security assurance: Security requirements, modeled by different security concerns, must individually guarantee assurance, i.e., if RBAC is modeled, there is a guarantee that each defined role will access exactly what is needed and no more. Further, the composition of security concerns must also be secure.

At the modeling level, related approaches are: [6] proposes an extension to UML that defines several new stereotypes towards formal security verification of elements (e.g., fair exchange, confidentiality, etc.); [7] defines a metamodel to generate security definition languages, an instance of which is SecureUML, a platform-independent language for role-based access control (RBAC); [1] defines an approach AuthUML that includes a process and a modeling language to express RBAC policies via use cases; [5] proposes a network enterprise framework using UML to represent RBAC requirements for the framework introduced in [15]; and, [4] extends UML with both RBAC and mandatory access control (MAC). All of these approaches model security requirements without considering the translation to a concrete implementation that preserves separation of concerns.

Two approaches focused on separation of concerns during development. In [11], the authors present an example of composition of access control behavior into an application via the aspect-oriented methods whereas [16] provides a general framework to incorporate security into software using aspect-oriented programming to manage authentication and intercept method calls to constrain them based on permissions. Both approaches emphasize composition of security behavior into an application, depending on higher-level languages to accurately represent security concerns: [11] emphasizes security in models, and [16] implements security using aspect-oriented code. Our approach advocates a more global vision of security comprised of a model for security (the role slice model) which is integrated with a mapping to aspect-oriented code.

My doctoral research proposes a composable security definition, assurance, and enforcement via a model-driven framework that preserves separation of security concerns from modeling through implementation, and provides mechanisms to compose these concerns into the application while maintaining consistency between design models and code. At modeling-time, separation of concerns (e.g., RBAC, MAC, delegation, authorization, etc.) is emphasized by defining concern-specific modeling languages. At an implementation-level, aspect-oriented programming (AOP) transitions security concerns into modularized code that enforces each concern. The material presented throughout the paper assumes the use of an underlying object-oriented language with aspect-oriented extensions, and infrastructure to implement the applications and support secure access to the public methods of classes, e.g., Java with AspectJ[14] or C++ with AspectC++[12].

2 The Proposed Framework

The proposed framework for composable security definition, assurance, and enforcement is given in Fig. 1, and augments the software process at the design and implementation stages. In the top half of the figure, the design stage is represented, including the *Main Application Design Model* (e.g., UML) that contains all of the domain-specific information about the application (i.e., business rules). In addition, this design model is augmented with *Security Concern Models* for alternative security features, such as access control (e.g., RBAC, MAC, delegation, etc.), auditing, and authorization. When designing an application, the software/security engineer can pick and choose, as independent concerns, their required security, e.g., a banking application might use RBAC for customers and MAC with 4 security levels for banking personnel. The Security Concern Models contains fine-grained units that are selectable, parameterizable, and composable to handle such a situation. The bottom half of the figure represents the mapping to the *Main Application Implementation*, which involves the coding of an application's classes, methods, etc., and associated *Security Aspect-Oriented Implementation*, which involves an automated mapping to AOP code that realizes the chosen Security Concern Models for the application. For this research, these mappings employ Java and AspectJ, respectively. Composition allows the AOP code for the chosen Security Concern Models (e.g., for RBAC, MAC, and 4 security levels) to be combined with the application. The main research tasks required to realize the framework are:

1. Identify a broad set of Security Concern Models (e.g., RBAC, MAC, delegation, authorization, parameters of security models, etc.) that are both quantifiable units and composable. The objective is to offer a wide range of security capabilities to software/security engineers. The main criteria to identify a composable concern is to determine if its properties can be expressed as formal method preconditions. For example, in our current research [8,9], RBAC can be composed into an application, by doing permission checking as a precondition of the methods that need access control.
2. Design a means to integrate the Security Concern Models into a design model (UML) to capture security requirements as part of the software process. This may involve extending existing UML capabilities, proposing new UML diagrams, and/or integrating with other security modeling techniques.

3. Develop a formal model to represent security and non-security concerns that captures a design state for use in static analyses of the security properties of the framework. The formal model is crucial to support security assurance and track potential security conflicts and inconsistencies.
4. Design formal rules that will govern the mapping of each Security Concern Model to AOP enforcement code, including the composition of multiple concerns and application code. For example, the conditions required to compose two security policies (RBAC and MAC), the interpretation of permissions acquired via inheritance (overload or override), and, the rules that delineate the impact on security concerns when application classes are added, modified, and/or deleted.

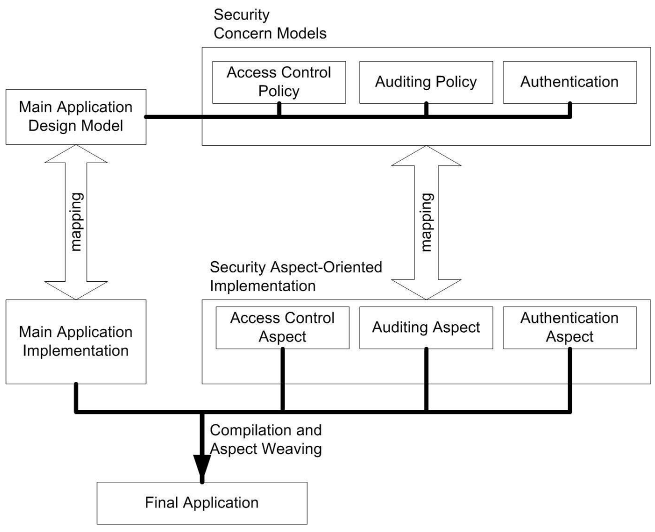

Fig. 1. Approach for Modeling Security.

3 Expected Contributions and Results

The central contribution expected from this research is a complete framework that integrates security with the software process, preserves the separation of security and non-security concerns, and yields applications that are the composition of application and enforcement code. Specific contributions include:

– Visual and non-visual modeling extensions to UML that represent and integrate all of the Security Concern Models into the software process.
– Strong assurance that the AOP code generated for every individual Security Concern Model, and for their composition with one another, is secure.

– A formal model to capture security and non-security application concerns, a design state, leveraged to prove assertions regarding security consistency and completeness of individual Security Concern Models and their composition.
– Detailed algorithms that map Security Concern Models (and their composition) into composable AOP enforcement code that preserves separation of concerns.

A software prototype is being built (see Section 4), and will be utilized for experimental validation of the research.

4 Research Status

As of this writing, the status of the research plan outlined above is as follow[1]:

1. Chosen RBAC as the first Security Concern Model, and in terms of UML, define roles whose positive permissions are a subset of the public methods of the application's class library, as reported in [8].
2. Proposed a new UML artifact, the *role-slice diagram*, that allows a software/security engineer to capture the Security Concern Model for RBAC [8]. The role slice provides an abstraction to collect information on the permissions of a role that cuts across all classes in an application.
3. Developed an initial formal model [9] for security and non-security concerns via a functional notation based on structural operational semantics [10].
4. Designed algorithms for mapping a role-slice diagram to AOP security enforcement code [8,9] via *model composition* [2] to manage role hierarchies.

Lastly, a prototype is being developed in collaboration with a fellow Ph.D. candidate and a team of M.S. students based on earlier work described in [8,9]. The prototype integrates the role-slice diagram as well as other security work on extending UML with RBAC/MAC [4]. The prototype has been implemented as a plugin of Borland's Together Control Center, and includes mapping role-slice information to AspectJ security enforcement code. We note that we are in the planning stage for transitioning this work into the Eclipse environment.

References

1. Khaled Alghathbar and Duminda Wijeskera. Consistent and complete access control policies in use cases. In Perdita Stevens, Jon Whittle, and Grady Booch, editors, *UML 2003 - The Unified Modeling Language. Model Languages and Applications. 6th International Conference, San Francisco, CA, USA, October 2003, Proceedings*, volume 2863 of *LNCS*, pages 373–387. Springer, 2003.
2. Siobhán Clarke. *Composition of object-oriented software design models.* PhD thesis, Dublin City University, January 2001.
3. Premkumar T. Devanbu and Stuart Stubblebine. Software engineering for security: a roadmap. In *Proceedings of the Conference on The Future of Software Engineering*, pages 227–239, 2000.

[1] it mirrors the tasks 1 through 4 described in Section 2

4. T. Doan, S. Demurjian, T.C. Ting, and C. Phillips. RBAC/MAC security for UML. In C. Farkas and P. Samarati, editors, *Research Directions in Data and Applications Security XVIII*, July 2004.
5. Pete Epstein and Ravi Sandhu. Towards a UML based approach to role engineering. In *Proceedings of the fourth ACM workshop on Role-based access control*, pages 135–143, 1999.
6. Jan Jürjens. UMLsec: Extending UML for secure systems development. In *Proceedings of the 5th International Conference on The Unified Modeling Language*, pages 412–425. Springer-Verlag, 2002.
7. Torsten Lodderstedt, David A. Basin, and Jrgen Doser. SecureUML: A UML-based modeling language for model-driven security. In *Proceedings of the 5th International Conference on The Unified Modeling Language*, pages 426–441. Springer-Verlag, 2002.
8. J. A. Pavlich-Mariscal, T. Doan, L. Michel, S. A. Demurjian, and T. C. Ting. Role Slices: A Notation for RBAC Permission Assignment and Enforcement. In *Proceedings of 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, 2005.
9. Jaime A. Pavlich-Mariscal, L. Michel, and Steven A. Demurjian. A Formal Enforcement Framework for Role-Based Access Control using Aspect-Oriented Programming. In Lionel Briand and Clay Williams, editors, *ACM/IEEE 8th International Conference on Model Driven Engineering Languages and Systems*, Montego Bay, Jamaica, 2005.
10. G.D. Plotkin. A Structural Approach to Operational Semantics. Technical Report DAIMI FN-19, CS Department, University of Aarhus, 1981.
11. Eunjee Song, Raghu Reddy, Robert France, Indrakshi Ray, Geri Georg, and Roger Alexander. Verifiable composition of access control features and applications. In *Proceedings of 10th ACM Symposium on Access Control Models and Technologies (SACMAT 2005)*, 2005.
12. Olaf Spinczyk, Andreas Gal, and Wolfgang Schröder-Preikschat. Aspectc++: An aspect-oriented extension to c++. In *Proceedings of the 40th International Conference on Technology of Object-Oriented Languages and Systems.*
13. Peri Tarr, Harold Ossher, William Harrison, and Stanley M. Sutton, Jr. N degrees of separation: multi-dimensional separation of concerns. In *Proceedings of the 21st international conference on Software engineering*, pages 107–119. IEEE Computer Society Press, 1999.
14. The AspectJ Team. The aspectj programming guide. http://dev.eclipse.org/viewcvs/indextech.cgi/ checkout /aspectj-home/doc/progguide/index.html, 2003.
15. Dan Thomsen, Dick O'Brien, and Jessica Bogle. Role based access control framework for network enterprises. In *Proceedings of 14th Annual Computer Security Application Conference*, pages 50–58, Phoenix, AZ, December 7-11 1998.
16. Bart De Win, Bart Vanhaute, and Bart De Decker. Security through aspect-oriented programming. In *Proceedings of the IFIP TC11 WG11.4 First Annual Working Conference on Network Security*, pages 125–138. Kluwer, B.V., 2001.