

# A Formal Approach to Designing Secure Software Architectures \*

Huiqun Yu, Xudong He, Yi Deng, Lian Mo  
School of Computer Science  
Florida International University  
Miami, FL 33199, USA  
{yhg|hex|deng|lmo01}@cs.fiu.edu

## Abstract

*Software architecture plays a central role in developing software systems that provide basic functionality and satisfy critical properties such as reliability and security. However, little has been done to formally model software architectures and to systematically enforce required properties. We aim to propose a formal approach to designing secure software architectures. We use the Software Architecture Model (SAM), a general software architecture model combining Petri nets and temporal logic, as the underlying formalism. Architecture design consists of the functionality part and the security part. Guidelines are proposed to design functionality of software architectures at both element level and composition level. Software security is enforced by stepwise refinement.*

## 1 Introduction

Software security has emerged as a foremost concern for modern information enterprise. Several well-known security system architectures and models, including CORBA, EJB, and DCOM, are cornerstones for designing scalable and flexible security systems. Despite these advances, however, how to analyze the design of security systems to ensure its consistency and integrity is still a largely open problem. There is lack of rigorous and systematic ways in the literature to assess and assure critical properties in architectural composition of security systems. Although formal verification of security protocols has received increasing attention in recent years [3, 8], these techniques are normally based on abstract computation models and are not concerned with composition or architecture of security systems. Many of these formal models or techniques are developed for a single security model and do not scale well.

---

\*Supported in part by the NSF under grants HRD-0317692 and CCR-0226763, and by NASA under grant NAG 2-1440.

To address the problem, we propose a formal approach to designing secure software architectures based on SAM [9]. In SAM, a software architecture is defined by a hierarchical set of compositions, each of which consists of a set of components, a set of connectors and a set of constraints to be satisfied by the interacting components.

Our research objectives are to provide a formal architectural model for security systems, and a formal method for security enforcement and analysis.

## 2 Research Description

Security system architecture design in SAM includes two parts. One is the functionality part, which deals with the overall structure of the software architecture, including the components, connectors, their hierarchical relationships, as well as the interfaces. The other is the security part, which handles security requirement modeling, specification, and enforcement.

There are two distinct levels of software architecture for functionality design in SAM, i.e. element level and composition level. In SAM, each element (either a component or a connector) is specified by a tuple  $\langle S, B \rangle$ .  $S$  is a property specification, written in temporal logic [7], that specifies the required properties of the element and  $B$  is a behavior model, defined by a Predicate Transition net (PrT net) [4], that defines the behavior of the element.  $S$  and  $B$  can be view as the specification and the implementation, respectively, as in many other software architecture models such as Wright [1]. Therefore, to model an element is essentially to write  $S$  and  $B$ . To define an element constraint  $S$ , we can either directly formulate the given user requirements or carry out a cause and effect analysis by viewing input ports as causes and output ports as effects. The general procedure to develop  $B$  includes the following steps.

1. Use all the input and output ports as places of  $B$ ;
2. Identify a list of events directly from the user requirements or through Use Case analysis [2];

3. Represent each event with a simple PrT net;
4. Merge all the PrT nets together through shared places to obtain  $B$ ;
5. Apply the transformation technique [6] to make  $B$  more structured and/or meaningful.

For composition level design, SAM supports both top-down and bottom-up system development approaches. The top-down approach is used to develop a software architecture specification by decomposing a system specification into specifications of components and connectors and by refining a higher-level component into a set of related sub-components and connectors at a low level. The bottom-up approach is used to develop a software architecture specification by composing existing specifications of components and connectors and by abstracting a set of related components and connectors into a higher-level component. Often both the top-down approach and the bottom-up approach have to be used together to develop a software architecture specification.

Security models are used to precisely describe the security relevant features of information systems. These models can be broadly categorized into *access control models* and *information flow models*. Access control models describe the protection features of information access. The components of an access control model include

- a set of subjects,
- a set of objects,
- an access matrix that maintains the protection state of the system, and
- a set of rules for changing the protection state of the system.

Information flow models deal with information flows that can be used to check whether any illegal flows can occur. In addition to those components for access control models, classes (or levels) for entities in information flow models are necessary.

We develop a formal way to embed the protection features of a software system in PrT nets. Each place in a PrT net represents a subject. The token contains interested attributes of its corresponding subject. Access matrix can be derived from the markings of places. The access control rules are coded as transitions and taken as operators on Petri nets. These rules provide a disciplined way to construct Petri net models, in which security policies are proved to be correctly enforced.

Three correctness criteria of SAM models are identified in [5], which include *element correctness*, *composition correctness*, and *refinement correctness*. Basically, to ensure

the correctness of a SAM model is to show that all the constraints are satisfied by the corresponding behavior models. There are broadly two approaches to correctness. One approach is an ad hoc design followed by verification. Several verification techniques have been established in SAM, including reachability tree technique, deductive proof technique, and structural induction technique [5]; The other approach is correctness by construction, which involves a series of correctness-preserving transformations from one specification to another, and the final specification will meet certain required properties. Our security enforcement technique follows the latter approach.

Our future research interests include multi-policy enforcement, modularity in policy representation, composition, design, and analysis tools.

## References

- [1] R. Allen and D. Garlan. A formal basis for architectural connection. *ACM Transactions on Software Engineering and Methodology*, 6(3):213–249, 1997.
- [2] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley Longman, Inc., 1999.
- [3] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Transactions on Computer Systems (TOCS)*, 8(1):18–36, 1990.
- [4] X. He. A formal definition of hierarchical predicate transition nets. In *Proceedings of the 17th International Conference on Application and Theory of Petri Nets, LNCS 1091*, pages 212–229. Springer-Verlag, 1996.
- [5] X. He and Y. Deng. A framework for developing and analyzing software architecture specifications in SAM. *The Computer Journal*, 45(1):111–128, 2002.
- [6] X. He and J.A.N. Lee. A methodology for constructing predicate transition net specifications. *Software-Practice and Experience*, 21(8):845–875, 1991.
- [7] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag, 1992.
- [8] S. Schneider. Verification authentication protocols in CSP. *IEEE Transactions on Software Engineering*, 24(9):741–758, 1998.
- [9] J. Wang, X. He, and Y. Deng. Introducing software architecture specification and analysis in SAM through an example. *Information and Software Technology*, 41:451–467, 1999.