Modular Security: Design and Analysis

Jie Ren
Talk for Advancement to Candidacy
June 2004
Outline

- Research Context:
  - How to design and analyze security of a software system composed of modules?
- Security
  - Confidentiality, Integrity, Availability
  - Policy, Model, Mechanism
  - Access Control Models
  - Information Flow Models
- Module types and connection mechanisms
- Survey framework
- Surveyed techniques
- Assessments and research issues
Disclaimer

✦ This is a software talk
  – It views security from the software perspective
✦ Limited addressing of security
  – Not covered: policy composition, trust management, …
  – Future research probably will address more of them
Research Context

- A system is composed of modules. Modules can be heterogeneous. A system has security property, so does a module.
- Given a set of modules, how can we design a system so it can be secure?
- Given a system of modules, how can we analyze its security?
Security: Basic Properties

- Confidentiality
  - No improper information disclosure
- Integrity
  - No improper information modification
- Availability
  - No improper denial of service
Security: Policy, Model, and Mechanism

- **Policy**
  - Goals to be achieved and rules to be enforced

- **Model**
  - Formal representation of policies
  - Models: access control, information flow, others

- **Mechanism**
  - Hardware/software used to implement policies
  - Tamper-proof, Non-bypassable, Small
Security: Access Control

Discretionary

- Access is based on identity of subject (principal, requestor), object (resource), and right (permission, privilege).
  - Lampson, 1971; Harrison-Ruzzo-Ullman, 1976;

- Access Control Matrix
  - Access Control List
  - Capability
Security: Access Control

Mandatory

- Multi Level Security (MLS)
- Confidentiality
  - Bell-LaPadula, 1975
  - No read-up, no write-down
- Integrity
  - Biba, 1977
  - No read-down, no write-up
Security: Access Control

Others

- Brewer-Nash, 1989
  - Chinese Wall
  - Dynamic mandatory control; dynamic separation of duty

- Clark-Wilson, 1987
  - Commercial settings
  - Authentication, audit, well-formed transactions, separation of duty

- Role-based Access Control (RBAC)
  - Ferraiolo-Kuhn, 1992; ANSI Standard, 2004
  - Role as an extra-level of indirection
  - Ease of management, roles hierarchy, timing and dynamism
Security: Information Flow Models

- Confidentiality (Secrecy) Model
- Covert Channels: storage and timing
- First: Non-Interference
  - Goguen-Meseguer, 1982
- Many following definition:
Components and Connections (Composition)

- Abstract Computation
  - Logic (conjunction); Trace (input/output); Process Algebra (common event)
- Module/Object/Component
  - Procedure call, event-based; Connector;
- Component-based Software Engineering (CBSE) Component
  - Procedure call; broker; container
- Common-Off-The-Shelf (COTS) Component
  - Custom connection
Framework of Survey

- Security Models
  - Discretionary Access Control
  - Mandatory Access Control
  - Information Flow Security

- Component Types
  - Abstract Computation
  - Module/Object/Component
  - CBSE Component
  - COTS Component

- Connection Mechanism

- Approach
  - Top-down
  - Bottom-up

- Formalism/Tools
  - Formalism
  - Automation
  - Tool
Outline

✓ Research Context:
  ✓ How to design and analyze security of a software system composed of modules?
✓ Security
  ✓ Confidentiality, Integrity, Availability
  ✓ Policy, Model, Mechanism
  ✓ Access Control Models
  ✓ Information Flow Models
✓ Module types and connection mechanisms
✓ Survey framework
  ✪ Surveyed techniques
  ✪ Assessments and research issues
Surveyed Techniques

- Formal Techniques
- Wrappers
- Agents
- Meta Object Protocol (MOP)
- Components
- General composition frameworks
- Aspects
- Architectural approaches
Surveyed Techniques

✓ Formal Techniques
★ Wrappers
★ Agents
★ Meta Object Protocol (MOP)
★ Components
★ General composition frameworks
★ Aspects
★ Architectural approaches
Composing Specifications, Albadi-Lamport, 1990

Defining Liveness, Alpern-Shneider, 1985

Transition, trace, property
- Systems and properties are sets of traces
- Safety and liveness property

Reasoning of composite behavior
- Composition: what can be composed
- Refinement: conjunction implies system
Application and Limitation

- Has been used to verify integrity
  - Formal Development Methodology, early 80s
  - Composability for Security Systems, late 90s
- Effective, but labor intensive
  - Theorem prover
- Inapplicable to confidentiality
  - Safety and liveness are sets of traces
  - Confidentiality are power sets of traces
  - Composition opens new chances of interaction and observation for leaking
Formal II: Information Flow Security

- Given a component with one property and a component with potentially different properties, when they are composed using one composition construct, what property will the composite system satisfy?

- Composition Construct: Product, Cascade, Feedback
Theories and Applications

★ Many unifying frameworks
  – Trace-based: Selective Interleaving Function, McLean, 1994
    ★ Take two traces and produce a third one
    ★ Can processes accept the same events?
  – Logic-based: MAKS, Mantel, 2002
    ★ Predicates on trace operations

★ Few real applications
  – No consensus, remote from real systems, primitive composition, difficult to build
# Summary of Formal Techniques

<table>
<thead>
<tr>
<th>Technology</th>
<th>Security Model</th>
<th>Component Type</th>
<th>Connection Mechanism</th>
<th>Approach</th>
<th>Formalism &amp; Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integrity Verification, like CSS</td>
<td>Access Control</td>
<td>Logic Formula</td>
<td>Refinement</td>
<td>Top-down</td>
<td>Logic + PVS</td>
</tr>
<tr>
<td>Trace-based Information Flow, like SIF</td>
<td>Information Flow Security</td>
<td>Trace</td>
<td>Product, Cascade, Feedback</td>
<td>Bottom-up</td>
<td>Trace</td>
</tr>
</tbody>
</table>
Surveyed Techniques

- Formal Techniques
  - Wrappers
- Agents
- Meta Object Protocol (MOP)
- Components
- General composition frameworks
- Aspects
- Architectural approaches
Types of Wrappers

- **Wrapper**
  - Perform pre and post processing
  - Agents and MOP are more complex wrappers

- **Levels of wrappers**
  - Application-level wrapper
  - Library function-level wrapper
  - System library-level wrapper
  - System call-level wrapper
Mediator, Hypervisor, and Generic SoftwareWrapper

- **Mediator**: Balzer and Goldman, 2000
  - Library function level, Windows
  - Binary patch, write-protection, injection in process creation

- **Hypervisor**: Mitchem et al., 1997

- **Generic Software Wrapper**: Fraser et al., 2000
  - State machine, pattern
  - Install, activate
  - Models support
Issues in Using Wrappers

- Level applied
  - applicability
- Information available
  - Context of decision
- Security property
  - Relying, augmenting, or replacing
- Supporting extension mechanism
- Portability
- Performance
  - Trust to reduce overhead
Surveyed Techniques

- Formal Techniques
- Wrappers
- Agents
- Meta Object Protocol (MOP)
- Components
- General composition frameworks
- Aspects
- Architectural approaches
Agents

- More knowledge, more complex, more cooperation, less regular structure
- Secure Access Wrapper, Dawson et al., 1998
  - Mapping between autonomous MLS
- NRL Workflow/Pump, Kang et al. 1998
  - Constructing MLS workflow from single level workflow using Pump
- JIF/Split, Myers et al. 2002
  - Partition source code for secure execution in distrusting hosts
Safebot

- Filman and Linden, 1996
- Ubiquitous, communicating, dynamically confederating, monitoring and controlling existing applications
- Framework: language, compiler, library
- Not implemented
Surveyed Techniques

- Formal Techniques
- Wrappers
- Agents
- Meta Object Protocol (MOP)
- Components
- General composition frameworks
- Aspects
- Architectural approaches
Meta Object Protocol

- Reflection
  - Smith, 1984; Maes, 1987
- Meta Object Protocol
  - Kiczales et al., 1991
- Process
MOP and Security

- The Actor Model
  - Base actor and meta actor
- Security Meta Object
  - Riechmann, 1997
  - Attach meta-objects to possible references
  - Roles of meta references, roles and domains
- Java MOP
  - Compile-time, load-time, proxy-based runtime, and VM-based run time
  - Impact on permission sets
Kava

- Welch and Stroud, 1999
- Bytecode rewriting
  - Load time,
  - Selective,
  - Type-safe
- Capability
  - Method, constructor, field, exception
  - Spec file
  - Non-bypassability
    - User defined class loader
    - System defined class loader
    - Merged base and proxy references
- Security
  - Access Control,
  - Clark-Wilson: field, method, log.
Surveyed Techniques

- Formal techniques
- Wrappers
- Agents
- Meta Object Protocol (MOP)
- Components
- General composition frameworks
- Aspects
- Architectural approaches
Components

- Computer Security Contract
  - Khan and Han, 2001
  - Required and ensured
  - Event-based negotiation
  - Active interface with active contracts

- cTLA
  - Hermann, 2003
  - Uses Temporal Logic of Action
  - No dynamic composition yet

- Issues
  - Decidability
  - Trustworthiness of specifications
Surveyed Techniques

- Formal Techniques
- Wrappers
- Agents
- Meta Object Protocol (MOP)
- Components
- General composition frameworks
- Aspects
- Architectural approaches
## Composition Frameworks

<table>
<thead>
<tr>
<th>Technique</th>
<th>Component</th>
<th>Composition</th>
<th>Other feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICARIS 2001</td>
<td>General</td>
<td>Virtual Interface, New Container, Re-Assembly</td>
<td></td>
</tr>
<tr>
<td>CRSS 2000</td>
<td>Low-level services, High-level services</td>
<td>Selection of service providers</td>
<td>Remote provider, Survivability</td>
</tr>
<tr>
<td>IDIAN 1999</td>
<td>Intrusion Detection Components</td>
<td>Events exchange, Producer-Consumer negotiation</td>
<td>Formally described negotiation protocol</td>
</tr>
<tr>
<td>PSF 2003</td>
<td>View with declarative specification</td>
<td>Dynamic composition</td>
<td>Monitoring for secure session</td>
</tr>
</tbody>
</table>

- **Appealing idea**
- **Drawbacks:** Components, connection (dynamic, security), assurance
Surveyed Techniques

- Formal techniques
- Wrappers
- Agents
- Meta Object Protocol (MOP)
- Components
- General composition frameworks
- Aspects
- Architectural approaches
Aspects

- From Aspect-Oriented Programming to Aspect-Oriented Software Development
  - Cross-cutting concern,
  - Aspect: advice and pointcuts

- Application to security
  - Aspect-Oriented Security Framework,
    - Shah and Hill, 2003, C programming
  - Feature Selection
    - de Bruin and van Vilet, 2002, requirements
DADO

- Wohlstadt, Jackson, Devanbu, 2003
- Aspect-Oriented Middleware
  - Adaptlet: A pair of a client and a server
  - Extends IDL with advice and request, “that”
  - Implemented as source or binary instrumentation on existing CORBA channel
- Security: injecting security checks
  - Example: contactAuthentic advice, check advice, register request
- Middleware, Client/Server, Static IDL
Component Virtual Machine

- Duclos, Estublier, and Philippe, 2002
- Combines container and AOP
  - Container based approach
    - Target environment, callback, user flexibility
  - AOP limitations
    - Source code, transformation vs. interpretation, compile time
- Utilizes MOP for user flexibility
- Aspect Description Language and Aspect User Language
  - So user can specify how to use aspects
  - Security: check, application, generation
- Callbacks, deployment support, user-defined aspect
## Summary of Aspect Techniques

<table>
<thead>
<tr>
<th>Technology</th>
<th>Security Model</th>
<th>Component Type</th>
<th>Connection Mechanism</th>
<th>Approach</th>
<th>Formalism &amp; Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-TOS/JAC</td>
<td>Access Control</td>
<td>Base + Aspect</td>
<td>Meta Object, Meta Class</td>
<td>Top-down, Bottom-</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>up</td>
<td></td>
</tr>
<tr>
<td>AOSF</td>
<td></td>
<td>Base + Aspect</td>
<td>Weave</td>
<td>Top-down</td>
<td>Weaver</td>
</tr>
<tr>
<td>DADO</td>
<td>Access Control</td>
<td>Adaptlet</td>
<td>Extended CORBA</td>
<td>Top-down Bottom-</td>
<td>Extended IDL; service and</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>up</td>
<td>request</td>
</tr>
<tr>
<td>CVM</td>
<td>Access Control</td>
<td>Deployable Component</td>
<td>Container-based interception; dynamic composition</td>
<td>Bottom-up</td>
<td>Aspect Description Language and Aspect User Language</td>
</tr>
</tbody>
</table>
Surveyed Techniques

- Formal techniques
- Wrappers
- Agents
- Meta Object Protocol (MOP)
- Components
- General composition frameworks
- Aspects
- Architectural approaches
Approaches without connectors

- **ASTER**
  - Bidan and Issamy, 1997
  - Among the first for specifying security requirements for components and form composition based on those requirements
  - Uses a module interconnection language
  - Specification for encryption and authentication choices
  - Access control policies: combine subjects and rules
  - Limitations: security primitives, spec match, lack of connector, compositions of compositions
Approaches without connectors, cont.

- **System Architecture Model**
  - Deng et al., 2003
  - Combines Petri nets and Temporal Logic
  - Top-down approach for verifying constraints on components
  - Essentially verification of safety

- **Object-Oriented Labeling**
  - Peter Herrmann, 2001
  - Extend standard object-orientation notations, adopt Common Criteria
  - Uses Myers’s labeling model
Approaches with connectors

- Connector Transformation
  - Spitznagel and Garlan, 2001
  - Problem: add Kerberos to RMI
  - Alternatives: modify application, modify generator
  - Solution: transformations on connectors
    - Transforming data, combining connectors, adding a role, adding/removing states, imposing a connector
  - Limitation: connector-specific transformations
Approaches with connectors, cont.

- **SADL**
  - Riemenschneider et al., since 1997
  - Continuous refinement proof
    - Security-preserving proof, checkable proof
  - Security: Distributed Transaction Processing with MLS security
    - Application, resource manager, transaction manager
    - Theory interpretation and faithful interpretation between levels: exact mapping
    - Manual proof
    - Lower-level proofs can reuse mapping and higher-level proofs
  - Design a lot, specify some, prove a little
# Summary of Architectural Approaches

<table>
<thead>
<tr>
<th>Technology</th>
<th>Security Model</th>
<th>Component Type</th>
<th>Connection Mechanism</th>
<th>Approach</th>
<th>Formalism &amp; Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object-Oriented Labeling</td>
<td>Information Flow Security</td>
<td>Object</td>
<td></td>
<td>Top-down</td>
<td>Decentralized Labeling; Graph Rewrite</td>
</tr>
<tr>
<td>ASTER</td>
<td>Access Control</td>
<td>Component</td>
<td>Component selection</td>
<td>Bottom-up</td>
<td>Logic</td>
</tr>
<tr>
<td>SAM</td>
<td>Access Control</td>
<td>Petri net</td>
<td>Petri-net composition</td>
<td>Top-down</td>
<td>Petri net and Temporal Logic</td>
</tr>
<tr>
<td>Connector Transformation</td>
<td>Secure Communication</td>
<td>Regular component</td>
<td>Transformed secure connector</td>
<td>Top-down</td>
<td>Transformations</td>
</tr>
<tr>
<td>SADL</td>
<td>Mandatory Access Control</td>
<td>Component</td>
<td>Security-preserving Transformation</td>
<td>Top-down</td>
<td>Logic, PVS</td>
</tr>
</tbody>
</table>
Assessments of Surveyed Techniques

- Formal techniques
  - Scalability and usability
- Wrappers
  - Mature; can be challenging in implementation
- Agents
  - Flexibility vs. applicability
- Meta Object Protocol (MOP)
  - Low-level implementations for flexibility
- Components
  - Need further investigation on security specifications
- General composition frameworks
  - Not well defined, overly ambitious
- Aspects
  - Abstraction, use of MOP and Components, description and enactment
- Architectural approaches
  - Software architecture needs support for security
Research Issues

- Foundations: why is it hard?
  - What kind of security cannot be easily modeled like functionality?
  - How can we bridge the gap between theory and practice?
- Security properties
  - Integrity, confidentiality, availability
  - How to describe the requirements of these properties for components and systems?
- Secure software architecture
  - Will an architectural approach succeed in providing security?
  - How can we make a software architecture secure?
  - Will a connector be a right place to enforce security?
- Description and enactment
  - What are the right mechanisms for description and enactment?
  - Security/Assurance vs. Flexibility/Generality
Research Plan

- Architecture-centered and connector-oriented
- Lightweight formal methods: logic
- Practical security models: advanced access control, trust
- Component specifications on security
- Compositions handled by connectors
- Implementation aids: wrappers, meta-object protocol, and aspects.
- Automatic tools: design, generation, analysis, visualization
- Validation: build and analyze real systems