Identifying Security Aspects in Early Development Stages

March 5th, 2008
*Takao OKUBO†‡ Hidehiko TANAKA‡
†Fujitsu Laboratories Ltd.
‡Institute of Information Security
Japan
Table of Contents

1. Motivation
   - Why aspect?

2. Related works

3. Proposed approach
   - Identifying security aspects in early stages

4. Evaluation

5. Summary
1. Motivation
Backgrounds

Problems with software security

- Insufficient security expertise
  - The root of all evil

- Low Security coverage
  - $\Rightarrow$ Vulnerability

- Low maintainability/reusability
  - $\Rightarrow$ Development cost escalation
AOSD (Aspect-Oriented Software Development)

- Suitable for Non-functional Requirements (NFR)

⇒ Silver bullet?

Our research: Attempts to verify the assumption
Aspect-Oriented Programming (AOP)

- Crosscutting concern/Dependency Injection (DI)
  A concern (aspect) is injected into other concerns (programs)
- Modularity, Coverage, Reusability
Ideal World with Aspects(1)

Solution for Security Expertise Issues

Programs (except Security)

dependency

Aspects (Security)

General Developer

Implements Functions except Security

Implements Only Security

Security Experts
Issues with AOP

Add Security Aspects for Completed Web Server Programs
- Forceful Browsing
- Session Attacks

Issues with Coverage and Reusability
- Low Detection Accuracy
  - ad-hoc, implementation
- Low Reusability
  - dependent on specific codes
Our Goal

Aspects should be analyzed, and designed in earlier development stages

Today’s presentation

- Analysis methods
  - How to identify security requirements
  - How to achieve sufficient security coverage
    - For finding pointcut-candidate
2. Related Works
Threat modeling (Microsoft)

- A famous threat analysis method
- Precise analysis with DFD
  - Much cost needed
  - Architecture must be detailed
- Unexpressed attackers
  - Difficult to identify threats caused by various types of attackers
Analyzing Aspects


- Methods for identifying aspects
- Insufficient reference to:
  - How to identify enough security concerns
  - Security coverage of aspects (All the threats must be covered)
Misuse Case ([Sindre00])

Figure from [Sindre00]
Advantages of Misuse Cases

- Visualized analysis
  - UML-style diagram
  - Easy to understand
- Correspondence between threats and measures
- Security measures for aspect-candidates
Issues of Misuse Cases

- Security expertise required for eliciting mis-actors & misuse cases
  - Data assets unexpressed
  - Different types of mis-actor unexpressed
- Difficulty for designing aspects
  - Specifying crosscutting points (Pointcuts)
  - Coverage
3. Proposed Approach
Proposed Approach

- Extension of misuse cases
  - Mis-actor type extension
  - Data asset extension
  - Misuse case endpoint extension

- Procedure of identifying aspects and pointcuts
Mis-actor Type Extension

Use cases

(a) Unauthorized Mis-actor

(b) Authorized Mis-actor (another person)

(c) Mis-Actor (same person as the actor)
Data Asset Extension

- Data asset description
  ⇒ Data-oriented threat analysis

CIA attributes can be set to assets

<<C>>
Data Asset

<<A>>
Use Case

Data Flow Direction
3 Types of Endpoint

- Use case
- Actor (client)
- Channel (ex. network)
Procedure in an Analysis Stage

1. Describe use cases
2. Add data assets
3. Identify threats
   - Adding mis-actors and misuse cases
4. Identify security measures
   - Adding measure use cases
5. Identify aspects and pointcuts
Identifying Aspects

1. Integrate the same kind of threats
2. Integrate the same kind of measures ⇒ Aspects
3. Identify pointcuts
Integration of Misuse Cases

- Search Products
- Buy Products
- Refer User Info.
- Abuse
- Prevents
- Authentication
- Prevents
- Integrate Misuse cases
- Integrate Measures
Identifying Pointcuts

- Search Products
- Buy Products
- Refer User Info.
- Authentication

Pointcut
Specifying Pointcuts

Specify the timing that the measure must be injected

- Before (the use case is executed)
  - Authentication

- Around
  - Encryption

- After
  - Logoff

- Not specified
4. Evaluation
Expressiveness(1)

Original Misuse Cases

Actor

Tamper

MITM

enables

Use Case
Expressiveness(2)

Extended Misuse Cases

Tamper

Spoof

MITM

Actor

Tamper

XSS

CSRF

<<C>><<I>>

Data

enables

enables

enables
Coverage

Application to web systems

- Typical threats & measures can be identified (Including Vulnerability with Programming)
  - Injection attacks
  - XSS
  - CSRF

- Aspects & pointcuts can be specified (at the use case level)
5. Conclusion
Conclusion

- Aspect may be good for security
  - not ad-hoc AOP.
  - Analysis in early stages needed
- Security requirements (aspects) identification with extending misuse cases
  - For easier threat-identification
  - Clarified correspondence between threats and measures
  - Methods for identifying crosscutting points
- Application to the web domain
  - Threats at programming are predictable
  - Able to patterning in the web domain
Future Works

- Security framework with aspects
  - UML+Java+AspectJ
- Developing security patterns
  - Security analysis patterns
  - Security design patterns
THE POSSIBILITIES ARE INFINITE
Coverage

Security is Injected into All the Places where it is needed
Reusability

Program A (existing)

Aspect can be reused for building a new program

Aspect A (existing)

Program B (new)

Aspect can be reused for building a new program
Requirements in the Analysis Stage

- All the potential threats must be identified
- Security measures (aspects) must be identified for all the threats
- Pointcut-candidates must be specified
• Designing and Maintain Crosscutting Points (for Coverage and Reusability)
Designing Security Aspects(2)

- Automatic Code Generation for More Coverage & Reusability
- Developing Security Design Pattern
Backgrounds of Backgrounds

Need for Programming Control

“Only necessary to use that library?”
“Only to program in that manner?”

Why Analysis/Design/Testing is important?

- Persons Make Programs
- Programmers obtain freedom and power
- Most users are not the programmers