Continuous Secure Software Development and Analysis

Sophie Schulz, Frederik Reiche, Sebastian Hahner and Jonas Schiffl
Problems

Security is not considered from beginning

- Violations are detected late
  → costly

- Hard to retrace decisions
  → later analyses run independent from previous decisions
  → changing situations leads to … ?
Problems

Security is hard to evaluate

- Hard to evaluate/ systematically check security requirements
  → often done with threat models & scenarios

- Security is an evolving risk
  → Security must be observed over time
  → Necessary changes should be easy to detect
Security is difficult

- Multiple Aspects/ Topics
  - Confidentiality, Integrity, Availability, Authenticity, …
  - Security models often contain only few aspects

- Intrinsically dependent
  - Security leaks lead to other security lacks
  - Attackers often reach their goal by a sequence of attacks
Related Work

Approaches

[Ryoo et al.]
1) Vulnerability-oriented: Expert interview
2) Pattern-oriented: Analyze design patterns regarding identified vulnerabilities
3) Tactic-oriented: Investigate handling of attacks

[Khan]
In every development stage: Stage has issues → later stages will have issues
Requirement phase: Misuse case analysis to verify requirements
Design phase: Use misuse cases and vulnerabilities to perform threat modeling
→ Adapt design
Coding phase: Tests with static analyses and code reviews
Idea

- Holistic framework for multiple security aspects
- Appliable over time & react to changing requirements & contexts
- Blackboard principle: PCM
- Connect security and architecture
- Base on fine-grained, underlying assumptions
**Vision**

### Definition

- **Requirements**
- **Security Catalogues**
- **System Context**

#### Design Decisions

- **refine**

#### Assumptions

- **link**

### Design

- **Architecture**

#### specify

- **Palladio Component Model**

### Implementation
Vision

Definition
- Requirements
- Security Catalogues
- System Context

Design Decisions → Assumptions → link

Design
- Architecture
- Palladio Component Model

Implementation

refine assumptions & design

specify

refine
Confidentiality Analysis

Confidentiality
- Information is not made available or disclosed to unauthorized individuals [ISO 27000]
- Often ensured by access control

Idea
- Refine confidentiality assumptions to access control policies
- Formulate policies as constraints
- Verify policies through other analyses
Vision

Definition
- Requirements
- Security Catalogues
- System Context

Design
- Design Decisions
  - Design Decisions \(\xrightarrow{\text{refine}}\) Assumptions
  - Architecture \(\xrightarrow{\text{specify}}\) Palladio Component Model
  - Palladio Component Model \(\xrightarrow{\text{link}}\) Architecture-based Analyses

Implementation
- Code
  - Code \(\xrightarrow{\text{implement}}\) Code-based Analyses

Design Decisions refine Assumptions and Design
Assumptions link Architecture-based Analyses
Design Decisions refine Assumptions & Design
Composition of architecture- and code-based analyses

- Some security aspects can only be verified on certain levels
  - I.e., correctness on code level
  - Others must base on assumptions
  - If assumptions are false, analysis results are false

- Idea
  - Compose static security analyses of system view and source code
  - Reduce failures through assumptions of analyzable aspects
  - Analyze assumed aspects (on architecture level) using code-based analyses
Code-based Analyses

Code-level Specification and Verification of Security

- Verification of underlying assumptions (of higher-level results)
- Verification of reusable building blocks

Idea
- Use formal verification to verify specification of components
- Use protocol verification for security properties between components
Example – Access Control

- **Architecture level**
  - Use access control to achieve confidentiality
  - How is the access control designed on architectural level?
  - → Multiple assumptions

- **Architecture- and code-based level**
  - What are underlying assumptions?
  - Role model is applied correctly?

- **Code-based level**
  - Verify implementation of (parts of the) role model
Benefits

- Overviewable security
  - Decisions & assumptions are explicit
  - Security patterns/mechanisms & assumptions are annotated
  - Results of analyses are traced back to PCM

- Different analyses
  - Combination of different aspects

- Security from beginning and to the end
  - Early analyses are possible
  - Later analyses can refine/verify the results of the previous ones
Benefits

- Threat models / Attack scenarios / Attack models
  → I.e., attack needs some assumptions
  → are these negated by the assumed security mechanisms?

- Risk management & Quantification
  → I.e., risk of breaking some assumption
  → Risk of breaking some security mechanisms
<table>
<thead>
<tr>
<th>Reference</th>
<th>Source</th>
</tr>
</thead>
</table>