Architectural Analysis for Security (AAFS)

Jungwoo Ryoo and Priya Anand, Penn State University
Rick Kazman, SEI/University of Hawaii

To appear in IEEE Security and Privacy
Architectural Analysis

• Structured way of discovering

  ▪ **Design decisions** in software
    ➢ Present or
    ➢ Absent

  ▪ Quality attribute goals of stakeholders
    ➢ Security,
    ➢ Modifiability,
    ➢ Performance,
    ➢ Usability,
    ➢ Etc.
Significance of Architectural Analysis

• During early design
  ▪ Recommended

• During maintenance
  ▪ After the system is built
    ➢ A basis for refactoring
  ▪ Disruptive
  ▪ Costly
  ▪ Risky
Motivations and Significance

• Not too many
  ▪ **Well established** architectural analysis methods
  ▪ Example
    ➢ Architectural Tradeoff Analysis Method (ATAM)

• Not to mention
  ▪ Architectural analysis method specializing in security

• Dire need for *Architectural Analysis for Security (AAFS)*
  ▪ Security: Costly and risky → dominant concern
Our Approach

• The use of design constructs
  ▪ Helps reason about security

• AAFS
  ▪ Contains
    ➢ Tactic-oriented Architectural Analysis (ToAA)
    ➢ Pattern-oriented Architectural Analysis (PoAA)
    ➢ Vulnerability-oriented Architectural Analysis (VoAA)
  ▪ Uses
    ➢ Interviews
Tactics

• Design Technique
  ▪ To satisfy a single quality attribute requirement

• Aha! moment
  ▪ Why not for *architectural analysis*?

• SATURN 2014
Security Tactics

• Useful *vocabulary*
  ▪ During architectural design and *analysis*
    ➢ For security

• Intentionally *abstract*
  ▪ To establish a baseline
    ➢ For further investigation
Security Patterns

• Well-known solutions to
  ▪ Recurring security problems

• Refined and instantiated from
  ▪ Security tactics

• Closer to code
Vulnerabilities

• Software Weaknesses
  ▪ Exploitation by attackers
  ▪ Code level

• Vulnerability databases
  ▪ Common Vulnerabilities and Exposures (CVE)
  ▪ Common Weakness Enumeration (CWE)

• Relationship with architectural solutions
  ▪ Missing tactic or pattern
CVE vs. CWE

• Security scenarios or test cases

• CVE
  ▪ Individual incident reports
  ▪ More than 70,000 and still counting

• CWE
  ▪ Categories of the incident report
  ▪ 940 entries
Our Approach Provides a Holistic View of Security

• The ultimate goal
  ▪ To identify
    ➢ The absence or presence of a design decision
      → ToAA and PoAA
    ➢ The misinterpretation or violation of a design decision in the source code
      → VoAA
Steps of Our Methodology

• Step 1
  ▪ Tactic-oriented Architectural Analysis (ToAA)

• Step 2
  ▪ Pattern-oriented Architectural Analysis (PoAA)

• Step 3
  ▪ Vulnerability-oriented Architectural Analysis (VoAA)
Case Study

• OpenEMR
  ▪ Electronic Medical Record (EMR) System
  ▪ Open Source
    ➢ Released in 2001
    ➢ 531,789 LOC
    ➢ Big user base

• Factors in choosing a subject
  ▪ Access to architect and source code
ToAA Phase

• Interview an architect
  ▪ Where
  ▪ How

• Identify design
  ▪ Rationale
  ▪ Assumptions

2. With respect to security, what are the approaches that you have taken to address this quality attribute?

<table>
<thead>
<tr>
<th>Tactic</th>
<th>How is it achieved?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detect Intrusion</td>
<td>- Use of logging</td>
</tr>
<tr>
<td></td>
<td>- There is no capability to detect specific intrusion attempts such as SQL injection</td>
</tr>
<tr>
<td>Detect Service Denial</td>
<td>- Not supported by OpenEMR</td>
</tr>
<tr>
<td>Verify Message Integrity</td>
<td>- Partially supported by OpenEMR by means of standardized library function calls specializing in sanitizing user inputs</td>
</tr>
<tr>
<td>Detect Message Delay</td>
<td>- Not supported by OpenEMR</td>
</tr>
<tr>
<td>Identify Actors</td>
<td>- Implemented as part of the OpenEMR business logic</td>
</tr>
<tr>
<td>Authenticate Actors</td>
<td>- Implemented as part of the OpenEMR business logic</td>
</tr>
<tr>
<td>Authorize Actors</td>
<td>- Implemented as part of the OpenEMR business logic</td>
</tr>
<tr>
<td>Limit Access</td>
<td>- Implemented as part of the OpenEMR business logic</td>
</tr>
<tr>
<td></td>
<td>- Use of database views and role-based access control</td>
</tr>
<tr>
<td>Limit Exposure</td>
<td>- Not supported by OpenEMR</td>
</tr>
<tr>
<td></td>
<td>- OpenEMR is not modular at all. Therefore, the impact of a compromise can quickly spread throughout the system</td>
</tr>
<tr>
<td>Encrypt Data</td>
<td>- Not supported by OpenEMR</td>
</tr>
<tr>
<td>Separate Entities</td>
<td>- Not supported by OpenEMR</td>
</tr>
</tbody>
</table>
PoAA Phase

• Relate ToAA results to Patterns
  ▪ ‘Verify message integrity’ ← ToAA

• Check tactic realization
  ▪ Intercepting Validator
    ➢ Verifies user inputs before they are used
    ➢ Performs filtering to all requests or user inputs
      ✓ According to validation rules
    ➢ Forwards full, partial, or no input to the target
      ✓ Depending on the validation results
VoAA Phase

• Relate PoAA results to CWE categories
  ▪ Ties the suspicion to a piece of code

• CWE entries related to
  ▪ ‘Verify message integrity’ tactic
  ▪ ‘Intercepting validator’ pattern

• CWE 89: Improper neutralization of special elements used in an SQL command
• CWE 87: Improper neutralization of alternate XSS syntax
OpenEMR Analysis Sample Results

• ToAA
  ▪ ‘Verify message integrity’
    ➢ Partially supported by
      ✔ Standard library functions for sanitizing user inputs

• PoAA
  ▪ No intercepting validator

• VoAA
  ▪ CWE 89: Ad hoc and incomplete coverage
  ▪ CWE 87: No coverage
Verification

- Vulnerability analysis by IBM AppScan
  - OpenEMR
    - 3.1.0
    - 4.1.2

- SQL injection
  - Improving but still problematic

- XSS
  - Highly problematic
Future Research

• More case studies
  ▪ Nuxeo

• Tactic realization ontology

• Mapping between patterns and CWE entries
Questions?