

Architectural Analysis for Security (AAFS)

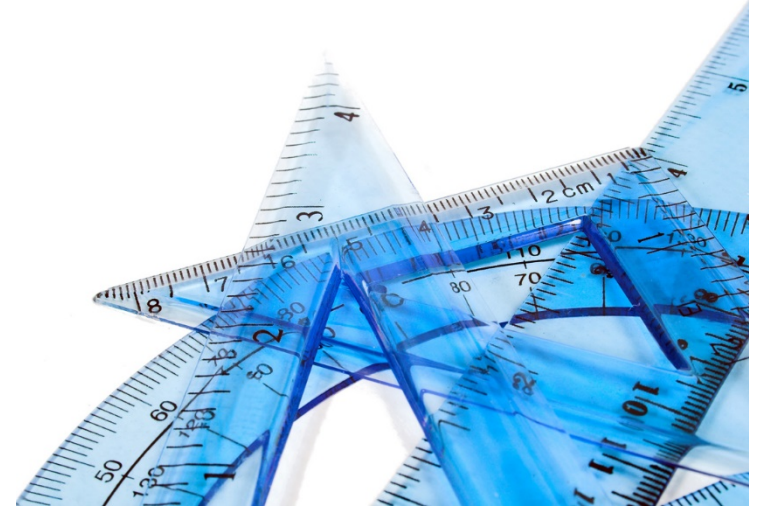
Jungwoo Ryoo and Priya Anand, Penn State University

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



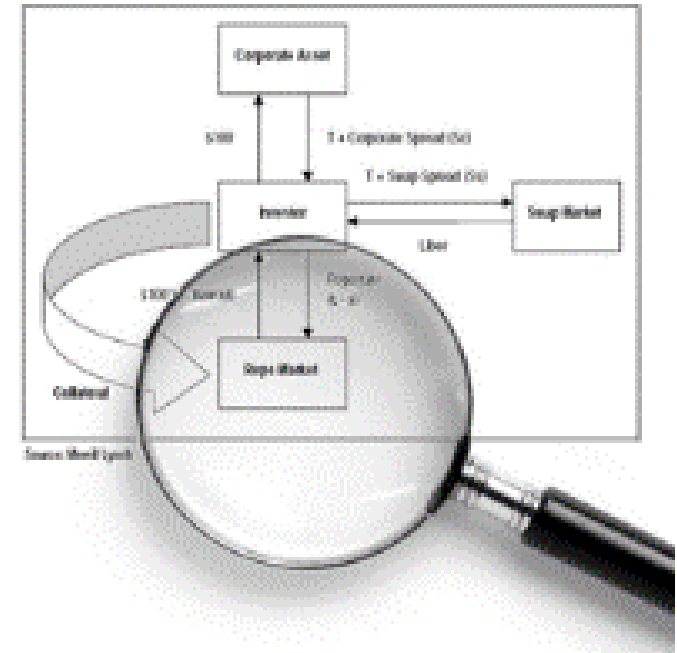
Motivations and Significance

- **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



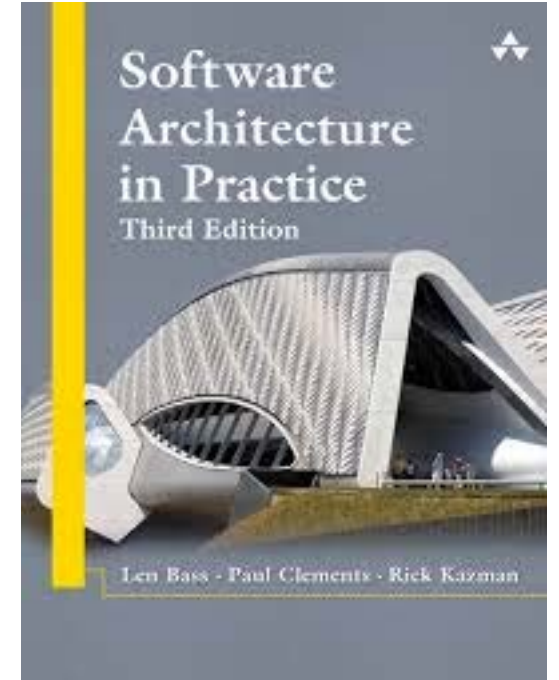
Our Approach

- 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



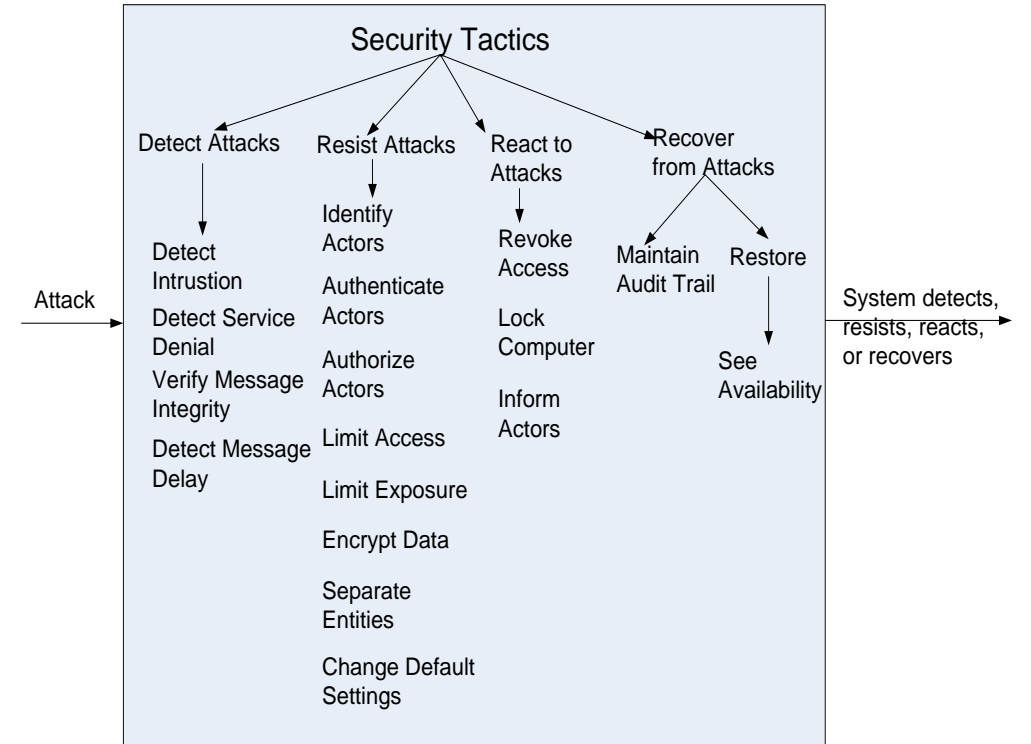
Tactics

- Design Technique
 - To satisfy a single quality attribute requirement
- Aha! moment
 - Why not for *architectural analysis*?
- SATURN 2014



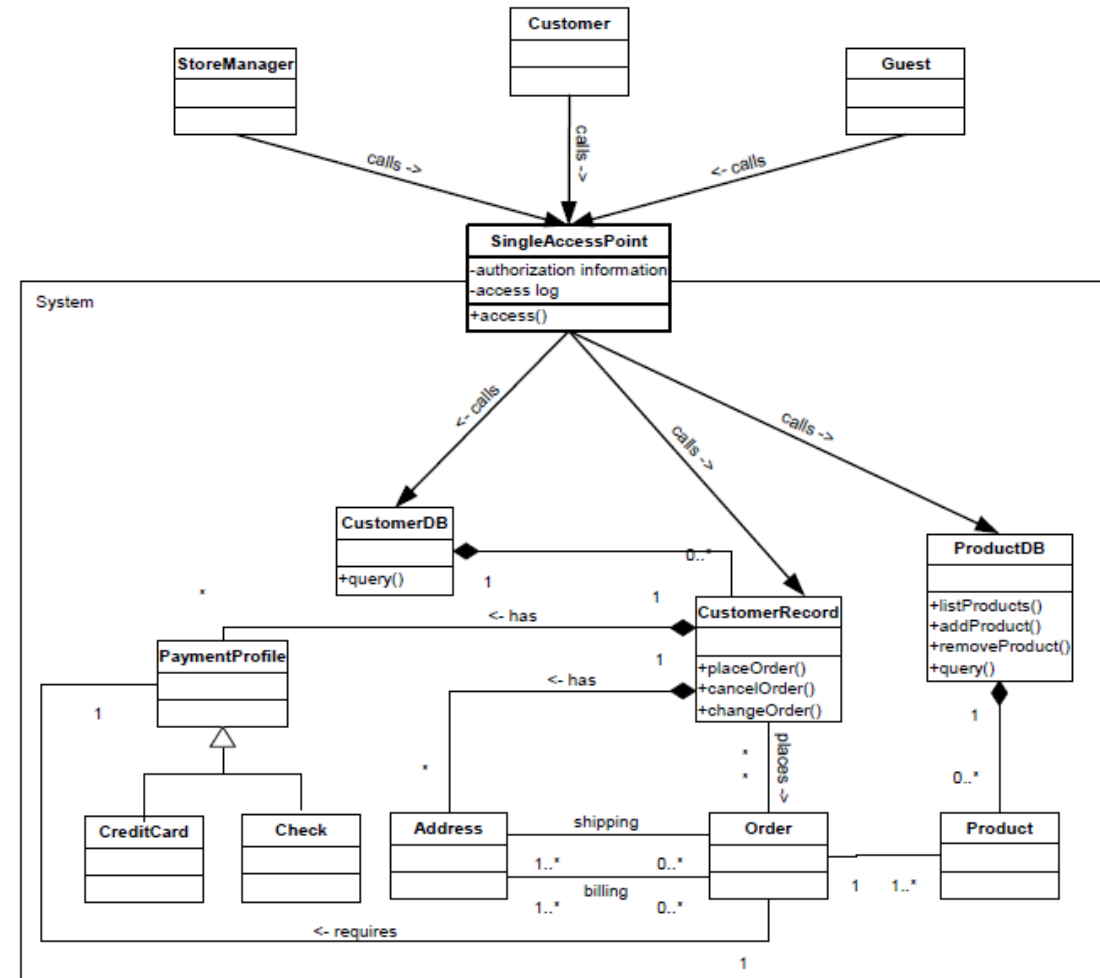
Security Tactics

- Useful *vocabulary*
 - During architectural design and *analysis*
 - For security
- Intentionally *abstract*
 - To establish a baseline
 - For further investigation



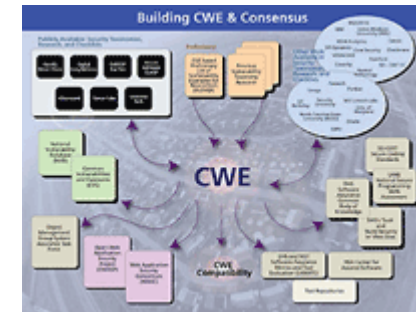
Security Patterns

- Well-known solutions to
 - Recurring security problems
- Refined and instantiated from
 - Security tactics
- Closer to code



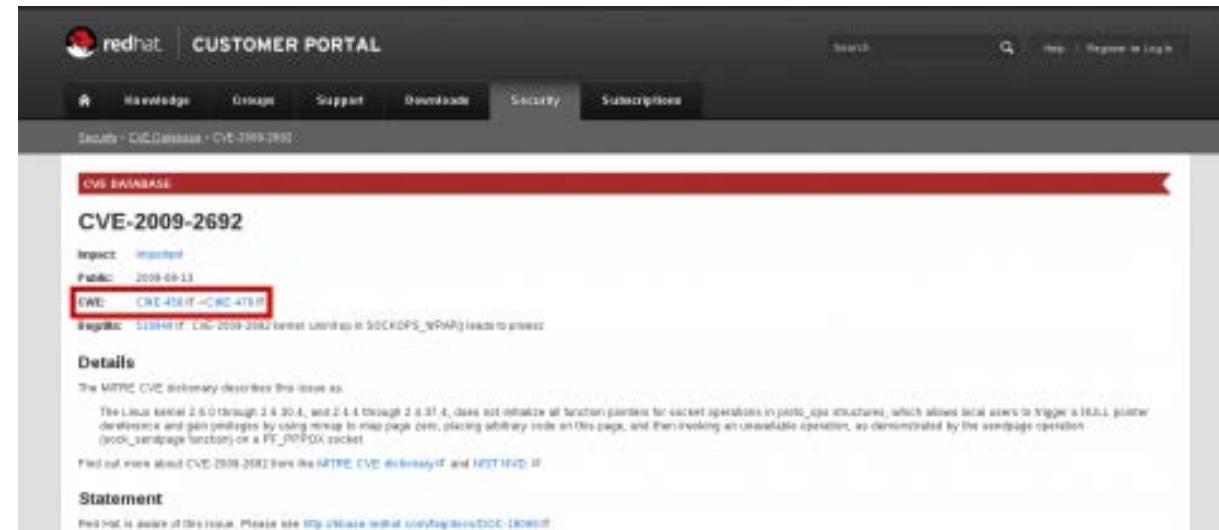
Vulnerabilities

- 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



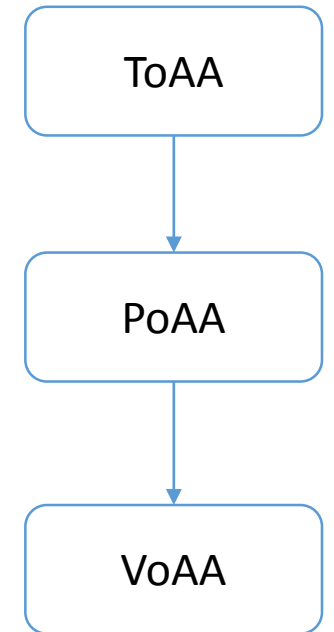
Our Approach Provides a Holistic View of Security

- 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



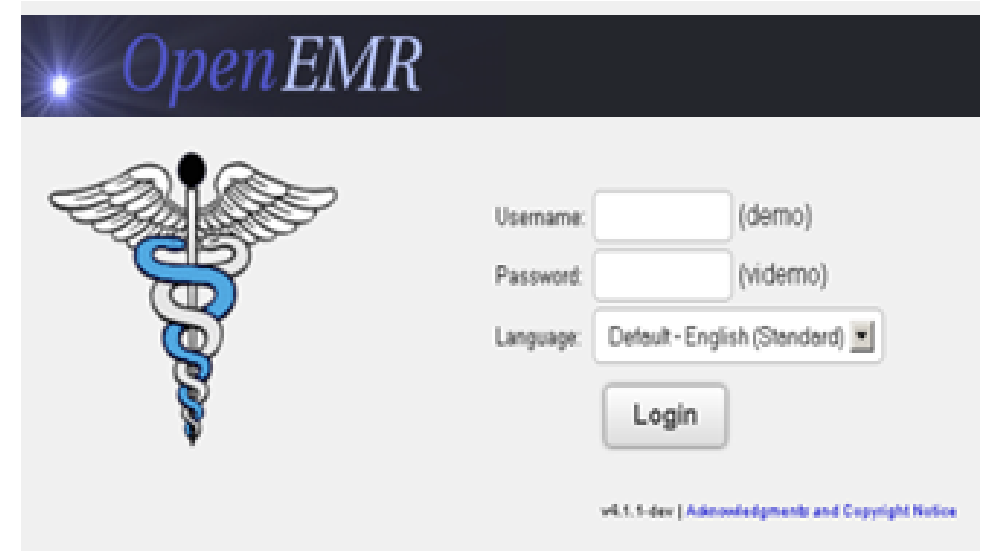
Steps of Our Methodology

- 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

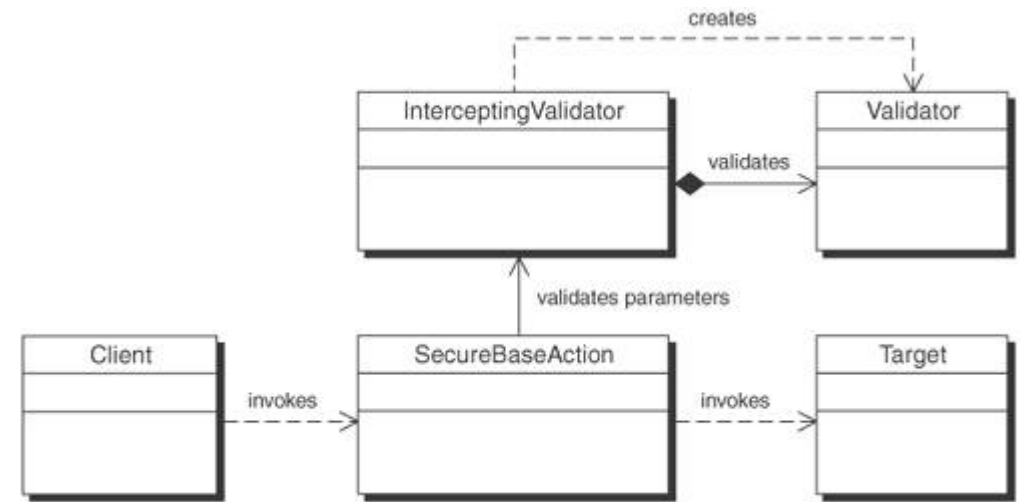
- Interview an architect
 - Where
 - How
- Identify design
 - Rationale
 - Assumptions

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

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

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

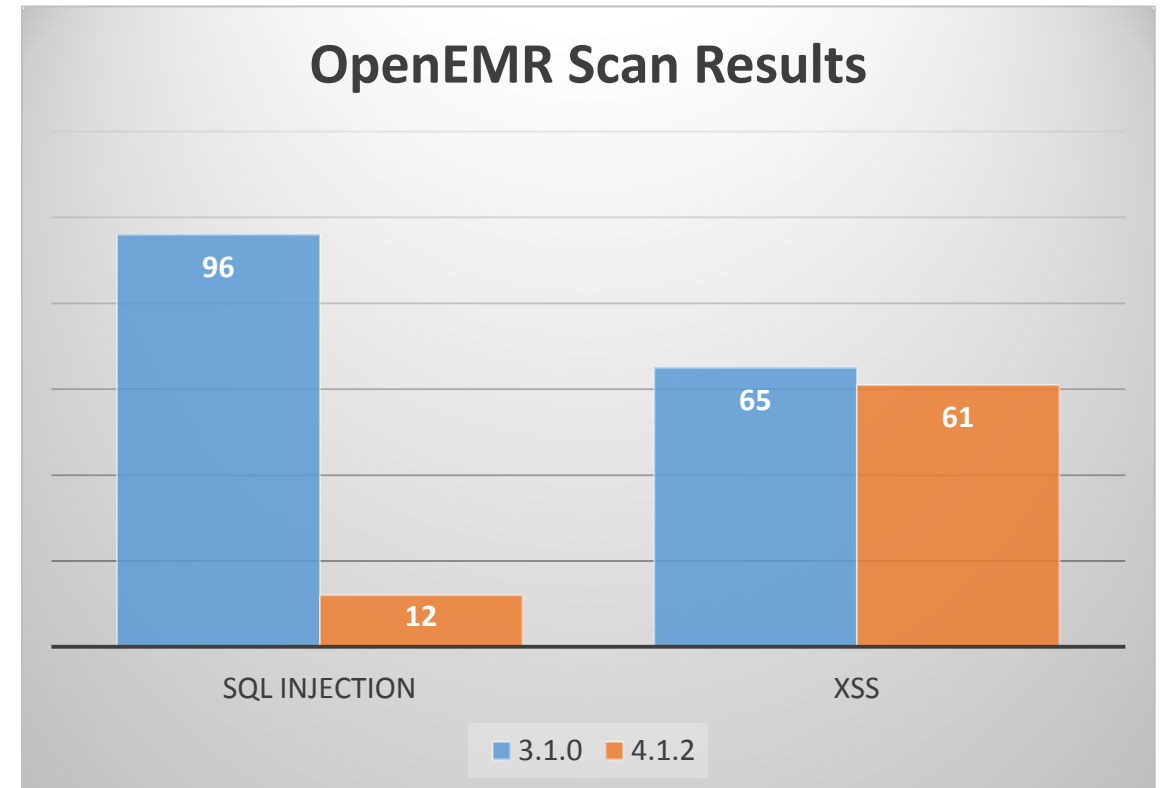
- ToAA
 - *'Verify message integrity'*
 - Partially supported by
 - ✓ Standard library functions for sanitizing user inputs
- PoAA
 - No intercepting validator
- VoAA
 - CWE 89: *Ad hoc* and incomplete coverage
 - CWE 87: No coverage



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?

