Approaching Security from an "Architecture First" Perspective

Rick Kazman, University of Hawaii
Jungwoo Ryoo, Penn State University
Humberto Cervantes, Universidad Autonoma Metropolitana-Itztapalapa

An Architectural Approach

Software security is a complex multi-dimensional problem, touching coding, design, operation, and policy.

Most software engineering effort goes into secure *coding*.

An Architectural Approach - 2

But secure coding is not enough.

Why?

1. Security is a “weakest link” phenomenon.
2. Secure coding practices are expensive.

An Architectural Approach - 3

We advocate an *architectural* approach to software security.

Specifically we advocate the use of security frameworks

- encapsulate best practices in design and coding

An Architectural Approach - 4.

What is the evidence for this advocacy?

Until now … nothing.

Three Case Studies

We now present three case studies.

We examine the effects of using a security framework on:

1. system quality, and
2. development efficiency.

Architectural Foundations

An architectural approach to software security relies on three related fundamental design concepts:

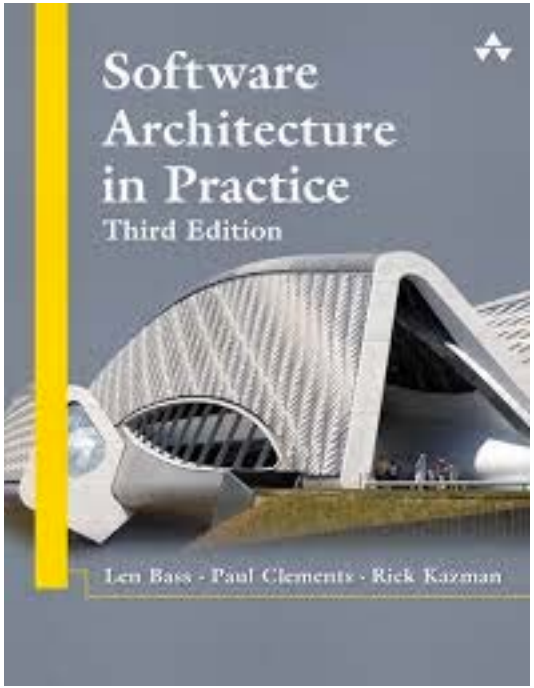
- tactics,
- patterns, and
- frameworks.

These concepts could apply to different quality attributes but here we focus on security.

Tactics

Architectural tactics are techniques that an architect can employ to achieve required quality attributes in a system.

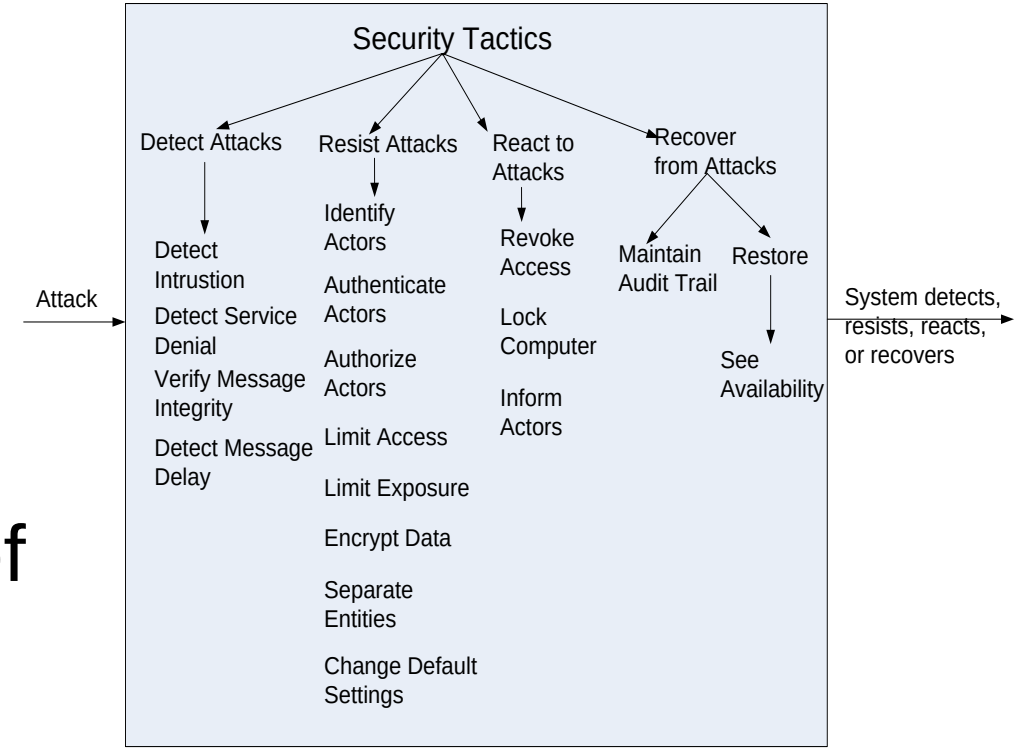
The tactics used here are taken from:



Security Tactics

Tactics provide a useful vocabulary for design and analysis.

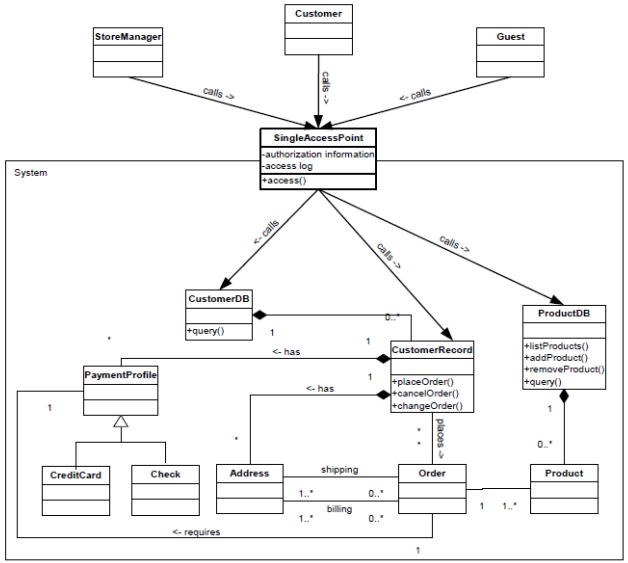
But realizing them in code involves lots of interpretation.



Security Patterns

There are a number of well-established security pattern catalogs.

Patterns help to structure a design, but they are difficult to correctly implement, maintain, and combine.



Security Frameworks

A framework is: a reusable software element that provides generic functionality addressing recurring concerns across a broad range of applications.

There are security frameworks for many languages and technology stacks.

Frameworks increase productivity, but often have a steep learning curve and "lock-in".

Case Studies

Given this wealth of design concepts, we were interested to understand:

- how architects approach security,
- how well these design approaches “perform” in terms of securing the system and reducing the cost of creating and maintaining a secure architecture.

Case Study Subjects

Organization name	Description	Case study	Frameworks used
CodeOne	Creator of a security framework in Korea	"ACME" web application	CodeOne Security Framework (“After”)
Quarksoft	Software consulting firm in Mexico City	Internal project management web application	ZK Spring Security
OpenEMR	Open source project	Electronic health records system	None

Case Study Protocol

1. Interview the architect regarding the approach to security, the size of the system, and the effort expended on security.
2. Scan the system to identify its vulnerabilities using *AppScan* from IBM.

Goal: explore tradeoff space between costs and benefits (effectiveness) of different approaches to security, and determine if there are optimal project strategies employing the approaches.

Interview Questions

1. What were your primary drivers (quality attributes for the system) and how important is security among them?
2. With respect to security, what are the approaches that you have taken to address this quality attribute?
3. How do you reason about tradeoffs?
4. How did you ensure that your programmers conform to the security approaches? (policies, inspections, etc.)
5. What percentage of project effort do you estimate goes into security without the use of a security framework? If using a security framework, what percentage of effort does this take?
6. Other comments

Example Answers

Tactic	How is it achieved?
Detect Intrusion	- Primarily enforced through the use of hardware firewalls - Spring Security also guarantees that a session comes from a single place
Detect Service Denial	- Covered by ZK - Use of hardware Firewall
Verify Message Integrity	- Covered by ZK. All requests are associated with a checksum and IDs. Most of the processing is done on the server.
Detect Message Delay	- Covered by ZK. When a session is created in ZK, many short-lived objects are created and each has a UID. The UID is verified by the framework so it would be hard to replicate these IDs.
Identify Actors	- Covered by Spring Security
Authenticate Actors	- Covered by Spring Security. All URLs are handled by Spring Security, transmission of content is a responsibility of ZK
Authorize Actors	- Covered by Spring Security
Limit Access	- Covered by Spring Security. The system runs over Tomcat, Spring Security overwrites the JAS standard from J2EE (nothing in particular was done, only roles were defined in the web.xml configuration file of the web server)

Tactic	How is it achieved?
Limit Exposure	- Not covered. Perhaps the fact that the application runs in an intranet?
Encrypt Data	- Use of HTTPS
Separate Entities	- Database server is physically separated, Identity Manager is also separated (it uses a Windows Active Directory).
Change Default Settings	- Not supported
Revoke access	- This can only be performed manually through the Active Directory.
Lock Computer	- Spring Security blocks the user if there are several attempts at accessing resources for which permissions are not granted.
Inform Actors	- Not supported
Maintain audit trail	- Several audit trails: Web server (audits web access), Spring Security (audits access to resources), ZK also creates logs.
Restore	- Not supported

Metrics Collected

- Vulnerability metrics were collected using *AppScan* which categorizes vulnerabilities as: High (H), Medium (M), Low (L), or Informational (I).
- Application size was measured using CLOC and MetricsReloaded
- Security effort was estimated by the interviewees.

Discussion

Our case studies represent three different security approaches, in terms of their architectural support for security (degree of adoption of frameworks):

- *Full adoption*: security framework used throughout the lifetime of the software, e.g. Quarksoft.
- *Partial adoption*: security framework is introduced in the middle of the lifetime, e.g. ACME “After”.
- *No adoption*: no use of any third-party security framework, e.g. OpenEMR, ACME “Before”.

Results

Case Study	ACME Before	ACME After	Quarksoft	OpenEMR
Approach followed	No adoption	Partial adoption (CodeOne Framework)	Full adoption (ZK and Spring Frameworks)	No adoption
Primary Development Language	JSP/HTML	JSP/HTML	Java	PHP
Size (KLOC)	7.93	8.55	16.56	255.6
Detected Security Issues (Vulnerabilities)	**H: 154** M: 50 L: 99 I: 224 Total: 527	**H: 0** M: 25 L: 99 I: 224 Total: 348	**H: 0** M: 0 L: 0 I: 1 Total: 1	**H: 8** M: 9 L. 475 I: 52 Total: 544
Analyzed URLs	756	756	24	3497
# of threat classes	9	8	1	8
Number of security tactics addressed	6/17	12/17	13/17	9/17
Number of security tactics addressed inside business logic	5/17	5/17	0/17	6/17
Estimated effort for security (% of total project effort)	20%	10%	3% (30% without frameworks)	20%

Inferences from the Results

1. The superiority of using security frameworks as an architectural approach, either through partial adoption or through full adoption.
2. The effort required for partial adoption is, however, significant when compared to the full adoption approach

Inferences from the Results - 2

Thus, we recommend the use of security frameworks from the early phases of the construction of a system (full adoption).

No big surprise: adopting a framework after the system has been built will clearly be more costly than doing so from the start.

Inferences from the Results - 3

Partial Adoption is a sub-optimal but common way of adopting security frameworks.

$\Rightarrow$ Most developers and architects worry about functionality first and security (and other quality attributes) later.

Conclusions

Why is it best to address security via frameworks?

1. while application developers may be experts in their domains, they are typically not security experts
2. even if developers are experienced in security, they should not write their own security controls
3. using a framework increases the likelihood that security controls will be applied consistently
4. delegating security issues to frameworks allows developers to devote their energy to application logic, increasing overall productivity

Questions?

We are actively looking for other case studies

- Interview with the architect
- *AppScan* vulnerability analysis

Feel free to contact us:

- kazman@hawaii.edu
- jryoo@psu.edu
- hcm@xanum.uam.mx