Lecture 16:
Development Process for Secure Software

Dr. Raimundas Matulevičius
email: rma@ut.ee

Development Processes
(Lecture outline)

- Emphasis on "building secure software" as opposed to "building security software"

- Major methodologies
  - Microsoft's Security Development Lifecycle
  - OWASP CLASP
  - Cigital's Security Touchpoints
Microsoft Security Development Cycle

- Core security training

- Establish security requirements
- Create quality gates/bug bars
- Perform security and privacy risk assessment
Microsoft Security Development Cycle


- Establish design requirements
- Perform attack surface analysis/ reduction
- Use threat modelling

- Use approved tools
- Deprecate unsafe functions
- Perform static analysis
Microsoft Security Development Cycle


Training
- Core security training

Requirement
- Establish security requirements
- Security and privacy risk assessment

Design
- Establish design requirements
- Analyse attack surface
- Threat modelling

Implementation
- Deprecate unsafe functions
- Static analysis

Verification
- Dynamic analysis
- Fuzz testing
- Attack surface review

Release
- Final security review
- Release archive

- Perform dynamic analysis
- Perform fuzz testing
- Conduct attack surface reviews

- Create an incident response plan
- Conduct final security review
- Certify release and archive
Microsoft Security Development Cycle


- Training
  - Core security training
- Requirement
  - Establish security requirements
  - Security and privacy risk assessment
- Design
  - Establish design requirements
  - Analyse attack surface
  - Threat modelling
- Implementation
  - Deprecate unsafe functions
  - Static analysis
- Verification
  - Dynamic analysis
  - Fuzz testing
  - Attack surface review
- Release
  - Final security review
  - Release archive

Response

- Execute incidence response plan

Open Web Application Security Project

OWASP

https://www.owasp.org/index.php/Main_Page

- Collect resources for Web applications
  - Top ten security flaws
  - Various security testing tools
  - Various security control means
    - e.g., code review guide

OWASP Appsec Tutorial Series

<https://www.youtube.com/playlist?list=PL8239DA448CC2BB7C>

- Basics
- SQL Injection
- Cross-site Scripting (XSS)
- HTTP Strict Transport Security

- CLASP – Comprehensive Lightweight Application Security Process
CLASP
https://www.owasp.org/index.php/Main_Page

• **Goal:**
  – move security concerns into the early stages of the software development lifecycle, whenever possible

• **Set of process pieces that can be integrated into any software development process**
  – Introduction to the Concepts behind CLASP to get started
  – Seven key Best Practices
  – High-level Security Services (authorisation, authentication, …)
  – Core Security Principles
  – Roles
  – Activities
  – Process engineering and roadmaps
  – Checklisted Coding Guidelines
  – Vulnerabilities that occur in source code
  – Searchable Vulnerability Checklist

---

**CLASP Best Practices**

• Institute awareness programs
• Perform application assessments
• Capture security requirements
• Implement secure development practices
• Build vulnerability remediation procedures
• Define and monitor metrics
• Publish operational security guidelines
CLASP Best Practices

• Institute awareness programs
• Perform application assessments
• Capture security requirements
• Implement secure development practices
• Build vulnerability remediation procedures
• Define and monitor metrics
• Publish operational security guidelines

• People should consider security to be an important project goal
• Train all team members
• Make people aware of security setting
• Institute accountability for security issues
• Appoint a project security officer
• Institute rewards for handling of security issues

CLASP Best Practices

• Institute awareness programs
• **Perform application assessments**
• Capture security requirements
• Implement secure development practices
• Build vulnerability remediation procedures
• Define and monitor metrics
• Publish operational security guidelines

• Security analysis of requirements and design
  – Threat modelling
• Source-level security review
• Security tests
CLASP Best Practices

• Institute awareness programs
• Perform application assessments
• **Capture security requirements**
• Implement secure development practices
• Build vulnerability remediation procedures
• Define and monitor metrics
• Publish operational security guidelines

• Treat security requirements same way as functional requirements
• Define security policy
• Identify attack surface
• Identify resources and trust boundaries
• Identify misuse cases
• Specify operational environment

CLASP Best Practices

• Institute awareness programs
• Perform application assessments
• Capture security requirements
• **Implement secure development practices**
• Build vulnerability remediation procedures
• Define and monitor metrics
• Publish operational security guidelines

• Annotate classes with security properties
• Apply principles of secure design
• Manage resources
• Manage contracts and interfaces
CLASP Best Practices

• Institute awareness programs
• Perform application assessments
• Capture security requirements
• Implement secure development practices
• Build vulnerability remediation procedures
• Define and monitor metrics
• Publish operational security guidelines

• Address reported security issues
• Manage security issue disclosure process

CLASP Best Practices

• Institute awareness programs
• Perform application assessments
• Capture security requirements
• Implement secure development practices
• Build vulnerability remediation procedures
• Define and monitor metrics
• Publish operational security guidelines

• Select metrics
• Collect data
• Evaluate results
CLASP Best Practices

- Institute awareness programs
- Perform application assessments
- Capture security requirements
- Implement secure development practices
- Build vulnerability remediation procedures
- Define and monitor metrics
- **Publish operational security guidelines**

- Build operational security guide
- Specify database security configuration

Seven Security Touchpoints


“All software projects produce at least one artifact: source code”
Seven Security Touchpoints

“All software projects produce at least one artifact: source code”

1. Code review (tools)  
2. Risk analysis  
3. Penetration test  
4. Risk-based security test  
5. Abuse analysis  
6. Security requirements  
7. Security attacks  
* External analysis

Code review (tools)

• Aim: catching implementation bugs early
• Tool helps to achieve good code coverage
• Aim for good, not perfect
Risk analysis

• Create description of architecture
  – Start with one page
  – Forest-level view

• Attack resistance
  – Use checklists of known attacks
  – Example: Microsoft STRIDE
    • Spoofing, Tampering, Repudiation, Info disclosure, Denial of service, Elevation of privilege

• Ambiguity analysis
  – Discover new risks
  – Find unclear parts in how the system works
  – Trust, data sensitivity, threat models

• Weakness analysis
  – Impact of external software dependencies
  – Platform (hardware, OS)
  – Frameworks
  – Called services

Combine risks and consider business impact
Rank risks
Find solutions

Penetration test

Attack on a system with the intention of finding security weaknesses, potentially gaining access to it, its functionality and data

• Use the source
  – Otherwise people send time on reverse-engineering system

• Apply business priorities
  – Logic flaw vs. XSS flaw
  – XSS is important if it contributes towards compromising business logic

• Use in-house QA department
  – They already know the system
  – Use tools and training to add security testing skills

• Test more than once

• Incorporate the findings back into development
Risk-based security test

• Test based on priorities
  – Architectural risks
  – Risks discovered during code review
• Test malicious input
  – Use fuzzing tool

Abuse analysis and Security requirement

• Security is not a set of features
• How system should react to illegitimate use
• Like use cases, but with malicious users
### Seven Pernicious Kingdoms

*Tsipenyuk et al., 2005*

<table>
<thead>
<tr>
<th>1. Input validation and Representation</th>
<th>5. Error Handling</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. API Abuse</td>
<td>6. Code Quality</td>
</tr>
<tr>
<td>4. Time and State</td>
<td></td>
</tr>
<tr>
<td></td>
<td>* Environment</td>
</tr>
</tbody>
</table>

### External analysis

- **Unfortunately**
  - Software architects, developers, and testers are largely unaware of the software security problems

- **Good news**
  - They acknowledge that security problems exists!

- **Bad news**
  - Barely **begun** to apply the security solutions
### Comparison

<table>
<thead>
<tr>
<th>Stages</th>
<th>SSDL</th>
<th>CLASP</th>
<th>Touchpoints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Education and awareness</td>
<td>1. Baseline education and advanced education</td>
<td>1. Base level and advanced education</td>
<td>1. No separate touchpoint, but recognises people should be sufficiently trained</td>
</tr>
<tr>
<td></td>
<td>2. Focus on awareness, knowledge of security engineering</td>
<td>2. Focus on all project roles</td>
<td>2. A knowledge management framework established to share security knowledge</td>
</tr>
<tr>
<td></td>
<td>3. Identifying and implementing</td>
<td>3. Emphasis on accountability</td>
<td>3. Stress on the creation and continuous execution of an improvement program</td>
</tr>
<tr>
<td></td>
<td>4. Tends to improve awareness by sharing all security artifacts</td>
<td>4. Tends to improve awareness by sharing all security artifacts</td>
<td></td>
</tr>
<tr>
<td>Project inception</td>
<td>1. Decision about the methodology, personnel, tools, and targeted security</td>
<td>1. Assignment of the security of the project and determining influence</td>
<td>1. Stress on the creation and continuous execution of an improvement program</td>
</tr>
<tr>
<td></td>
<td>2. Motivation plan by institutional accountability and by means of reward</td>
<td>2. Motivation plan by institutional accountability and by means of reward</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. Metric definition, collection and management</td>
<td>3. Metric definition, collection and management</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. Organisational policy management</td>
<td>4. Organisational policy management</td>
<td></td>
</tr>
<tr>
<td>Analysis and requirements</td>
<td>1. Use scenarios as the means in threat modelling</td>
<td>1. Identifies threats and vulnerabilities</td>
<td>1. Touchpoint dedicated to threat modelling based on above cases</td>
</tr>
<tr>
<td></td>
<td>2. Uses threat modelling</td>
<td>2. Identifies threats and vulnerabilities</td>
<td>2. Extra security requirements identified based on laws and regulations, commercial considerations and contractual obligations</td>
</tr>
<tr>
<td></td>
<td>3. Identifies threats and vulnerabilities</td>
<td>3. Identifies threats and vulnerabilities</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. Tends to reduce attack surface (by reduction of privileges)</td>
<td>4. Tends to reduce attack surface (by reduction of privileges)</td>
<td></td>
</tr>
<tr>
<td>Architectural and detailed design</td>
<td>1. Supports threat modelling</td>
<td>1. Supports threat modelling</td>
<td>1. Main focus on threat modelling</td>
</tr>
<tr>
<td></td>
<td>2. Focuses on operational environment</td>
<td>2. Focuses on operational environment</td>
<td>2. Includes threat identification and risk assessment</td>
</tr>
<tr>
<td></td>
<td>3. Assesses user privacy</td>
<td>3. Assesses user privacy</td>
<td>3. Tends to remove ambiguity</td>
</tr>
<tr>
<td></td>
<td>4. Tends to reduce attack surface (by reduction of privileges)</td>
<td>4. Tends to reduce attack surface (by reduction of privileges)</td>
<td></td>
</tr>
<tr>
<td>Implementation and testing</td>
<td>1. Provides secure coding guidelines</td>
<td>1. Provides secure coding guidelines</td>
<td>1. Big emphasis on security testing – three out of seven touchpoints deals with testing</td>
</tr>
<tr>
<td></td>
<td>2. Encourages both using automated tools for verification and manual code inspection</td>
<td>2. Encourages both using automated tools for verification and manual code inspection</td>
<td>2. Stress on importance of risk-based testing</td>
</tr>
<tr>
<td></td>
<td>3. Focus on black box testing</td>
<td>3. Focus on black box testing</td>
<td>3. Emphasis on code review using automated tools</td>
</tr>
<tr>
<td>Release, deployment and support</td>
<td>1. Focus on response plan (i.e., where, when vulnerability is determined)</td>
<td>1. Requests stakeholder to sign the code, so to provide a way to validate the origin and integrity of the product</td>
<td>1. Limited support in this activity (fine tuning access controls, configuring the monitoring and logging)</td>
</tr>
</tbody>
</table>

**SSDL**
- Education and awareness
- Project inception and release
- Deployment and support activities

**OWASP CLASP**
- Project inception and release
- Deployment and support activities
- Analysis and requirements
- Architectural and detailed design activities

**Seven Touchpoints**
- Implementation and testing
  - e.g., three out of seven touchpoints are related to the testing activities.
- Analysis and requirements
- Architectural and detailed design

[De Win et al., 2009]
Message to take home

- Emphasis on „building secure software“ as opposed to „building security software“

- Major methodologies
  - Microsoft's Security Development Lifecycle
  - OWASP CLASP
  - Cigital's Security Touchpoints

Final slide

- Don't do anything just because somebody else does
- Apply the scientific method
  - „a method of procedure that has characterized natural science since the 17th century, consisting in systematic observation, measurement, and experiment, and the formulation, testing, and modification of hypotheses.“

Creativity – your key to secure software