Outline

➢ Background
  • Previous studies on secure software development costs
  • Proposed rating scale
  • Data collection and Initial results of a Wide-band Delphi
  • Next steps
CyBoK

Software Security
Known categories of programming errors resulting in security bugs, & techniques for avoiding these errors—both through coding practice and improved language design—and tools, techniques, and methods for detection of such errors in existing systems.

Secure Software Lifecycle
The application of security software engineering techniques in the whole systems development lifecycle resulting in software that is secure by default.

Source: https://www.cybok.org/
Secure Software Development (Touchpoints)

Software Security Touchpoints

Software Security

Engineering software that continues working under malicious attack [McGraw, 2006]

Many issues faced in computer security today are rooted in our approach to developing software and systems [Heitzenrater, 2016]

Software defects have security ramifications [McGraw, 2006]

Security is an emergent property of a software system. There is no single addition of a feature that can make a software secure [McGraw, 2006]
Security Objectives and Requirements

• Security objectives establish confidentiality, integrity and availability requirements

• Functional
  • Security Features
  • Security Objectives -> Security Functional Components (CC)

• Non-functional (a quality requirement)

• In practice most software systems do not have explicit security objective/requirement

• Software security if often about avoiding known classes of bugs that enable attacks
Outline

• Background
  ➢ Previous studies on secure software development costs
• Proposed rating scale
• Data collection and Initial results of a Wide-band Delphi
• Next steps
# Sources of Cost (extracted from literature)

<table>
<thead>
<tr>
<th>Source</th>
<th>Papers</th>
<th>Source</th>
<th>Papers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perform Security Review</td>
<td>21</td>
<td>Perform Security Training</td>
<td>6</td>
</tr>
<tr>
<td>Apply Threat Modeling</td>
<td>18</td>
<td>Improve Development Process</td>
<td>5</td>
</tr>
<tr>
<td>Perform Security Testing</td>
<td>16</td>
<td>Perform Penetration Testing</td>
<td>5</td>
</tr>
<tr>
<td>Apply Security Requirements</td>
<td>11</td>
<td>Achieve Security Level</td>
<td>3</td>
</tr>
<tr>
<td>Apply Security Tooling</td>
<td>11</td>
<td>Document Technical Stack</td>
<td>3</td>
</tr>
<tr>
<td>Implement Countermeasures</td>
<td>9</td>
<td>Security Experts, Security Groups, Security Master</td>
<td>3</td>
</tr>
<tr>
<td>Fix Vulnerabilities</td>
<td>9</td>
<td>Track Vulnerabilities</td>
<td>3</td>
</tr>
<tr>
<td>Apply Secure Coding Standards</td>
<td>8</td>
<td>Functional Features</td>
<td>2</td>
</tr>
<tr>
<td>Apply Data Classifications Scheme</td>
<td>7</td>
<td>Hardening Procedures</td>
<td>2</td>
</tr>
<tr>
<td>Publish Operations Guide</td>
<td>7</td>
<td>Security by Design Paradigm</td>
<td>1</td>
</tr>
</tbody>
</table>

- **SWSec Practices**
- **Other Sources**
## Approaches to Estimating Costs of SWSec

<table>
<thead>
<tr>
<th>Approach</th>
<th>Additional Cost</th>
<th>Productivity Range</th>
<th>Source</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>COCOMO II security extension</td>
<td>0.94 (Low) 1.02 (Nominal) 1.27 (High) 1.43 (Very High) 1.75 (Extra High)</td>
<td>1.86</td>
<td>Expert estimation</td>
<td>Not validated</td>
</tr>
<tr>
<td>COSECMO</td>
<td>0% (Nominal) 20% to 80% (EAL 3 - High) 50 to 200% (EAL 4 - Very High) 125% to 500% (EAL 5 - Extra High) 313% to 1250% (EAL 6 - Super High) 781% to 3125% (EAL 7 - Ultra High)</td>
<td>31.25</td>
<td>Expert estimation with two inputs provided by a Commercial Company</td>
<td>Not validated</td>
</tr>
<tr>
<td>Weapon systems development cost model (COCOMO II based)</td>
<td>1.0 (Low or Nominal) 1.87 (High)</td>
<td>1.87</td>
<td>Expert estimation and 73 data points</td>
<td>Cross validation</td>
</tr>
<tr>
<td>Secure OS software cost model (COCOMO II based)</td>
<td>1 (Nominal) 1.25 to 1.5 (High) 1.75 to 2.0 (Very High) 2.0 to 2.75 (Extra High) 3.0 to 3.75 (Super High)</td>
<td>3.75</td>
<td>Expert estimation</td>
<td>Case study</td>
</tr>
<tr>
<td>FPA security extension (from General System Characteristics)</td>
<td>0 to 5% increase in the function points size of the project</td>
<td>1.05</td>
<td>Practices from survey with developers</td>
<td>Not validated</td>
</tr>
</tbody>
</table>
Practices Usage (from practitioners)

- Apply Secure Coding Standards
- Apply Security Tooling
- Track Vulnerabilities
- Apply Security Requirements
- Perform Security Testing
- Improve Development Process
- Document Technical Stack
- Perform Security Review
- Apply Threat Modeling
- Apply Data Classification Scheme
- Perform Penetration Testing
- Publish Operations Guide
- Perform Security Training

Percentage usage:
- Daily
- Weekly
- Monthly
State of Art versus Practice

• Sources of cost
  • Practices found in literature are applied in industry, i.e., literature and survey are consistent

• Estimation methods
  • Academia: COCOMO-based models
  • Industry: expert estimation

• Added costs for security
  • For rating = High:
    • COCOMO-based models -> 20% to 80% added effort
    • Survey results for New Development projects -> 31.6% average, 25% median (risk level 4)
      • Level 1: low likelihood of attack and low impact of attack
      • Level 5: high likelihood and high impact of attack
Outline

• Background
• Previous studies on secure software development costs
  ➢ Proposed rating scale
• Data collection and Initial results of a Wide-band Delphi
• Next steps
Security Driver for COCOMO III

• Based on software security practices
  • Found in literature
  • Usage confirmed in survey

• Degrees of applying security practices based on existing studies
  • Building Security In Maturity Model (BSIMM) v10
    • Published in 2019 with data observed from 122 firms
  • OWASP Software Assurance Maturity Model (SAMM) v1.5
    • An open framework to help organizations formulate and implement a strategy for software security
    • Published in 2017 with contributions from industry

• Rating scale will provide a measure of the level of secure software development
### Practices Consolidation

<table>
<thead>
<tr>
<th>Practices</th>
<th>BSIMM</th>
<th>SAMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apply Secure Coding Standards</td>
<td>Standards coverage</td>
<td>Basic ISO/IEC 25010:2011, moderate, comprehensive (proper use of APIs, memory sanitization, cryptography). Basic testing (simple edge cases and boundary conditions)</td>
</tr>
<tr>
<td>Perform Security Testing</td>
<td>Testing coverage and coverage</td>
<td>Basic use of static code analysis tools to identify security defects.</td>
</tr>
<tr>
<td>Apply Security Tooling</td>
<td>Tools usage</td>
<td>Basic tool configuration, customized with tailored roles, able to detect malicious code. Detailed technical stack documentation with third-party components identified and assessed based on security risks.</td>
</tr>
<tr>
<td>Perform Security Review</td>
<td>Review coverage and coverage</td>
<td>Ad-hoc security features code review.</td>
</tr>
<tr>
<td>Track Vulnerabilities (Development Time)</td>
<td>Resolution coverage</td>
<td>Critical vulnerabilities, high-risk vulnerabilities, moderate-risk vulnerabilities, low-risk vulnerabilities.</td>
</tr>
<tr>
<td>Improve Software Development Process</td>
<td>Improvement frequency</td>
<td>End of project, each release, each iteration.</td>
</tr>
<tr>
<td>Perform Penetration Testing</td>
<td>Penetration testing frequency</td>
<td>Before shipping, for each release, periodic.</td>
</tr>
<tr>
<td>Document Technical Stack</td>
<td>Control security of third-party components</td>
<td>Basic (identify and keep third-party components up-to-date on security patches), moderate (assesses third-party components risk).</td>
</tr>
<tr>
<td>Apply Data Classification Scheme</td>
<td>Data classification scheme</td>
<td>Simple classification (low-risk data), moderate classification (medium-risk data), complex classification (high-risk data).</td>
</tr>
<tr>
<td>Perform Security Training</td>
<td>Training level and coverage</td>
<td>General awareness, role-specific, advanced role-specific, customized with company data/knowledge, security certification.</td>
</tr>
<tr>
<td>Publish Operations Guide</td>
<td>Guiding coverage</td>
<td>Basic (critical security information for deployment), moderate (procedures for typical application alerts), thorough (formal operational security guide).</td>
</tr>
</tbody>
</table>
## Summarized Practices

<table>
<thead>
<tr>
<th>General Characteristic</th>
<th>Basic</th>
<th>Moderate</th>
<th>Extensive</th>
<th>Rigorous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Characteristics</td>
<td>High</td>
<td>Very High</td>
<td>Extra High</td>
<td>Ultra High</td>
</tr>
<tr>
<td>Design Summary</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Secure Coding and</td>
<td>Basic vulnerabilities applicable to the software will be prevented with secure coding standards and/or detected through basic use of static analysis tools.</td>
<td>Known and critical vulnerabilities applicable to the software will be prevented with secure coding standards and/or detected through routine use of static analysis tools.</td>
<td>Extensive list of vulnerabilities and weaknesses applicable to the software will be prevented with secure coding standards and/or detected through extensive use of static analysis and black-box tools.</td>
<td>Very extensive list of vulnerabilities and weaknesses applicable to the software will be prevented with secure coding standards and/or detected through rigorous use of static analysis and black-box security testing tools with tailored rules. Employ formal methods in coding.</td>
</tr>
<tr>
<td>Security Tools Summary</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security Verification</td>
<td>Basic adversarial testing and security code review. Basic penetration testing. Security V&amp;V activities conducted within the project.</td>
<td>Moderate adversarial testing and security code review. Routine penetration testing. Security V&amp;V activities conducted by an independent group.</td>
<td>Extensive adversarial testing and security design/code review. Frequent and specialized penetration testing. Security V&amp;V activities conducted by an independent group at the organizational level.</td>
<td>Rigorous adversarial testing and security design/code review. Exhaustive deep-dive analysis penetration testing. Use of formal verification and custom developed V&amp;V tools. Security V&amp;V activities conducted by an outside certified company.</td>
</tr>
</tbody>
</table>
## One-Line Summary

<table>
<thead>
<tr>
<th>None/Ad-hoc</th>
<th>Basic</th>
<th>Moderate</th>
<th>Extensive</th>
<th>Rigorous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nominal</td>
<td>High</td>
<td>Very High</td>
<td>Extra High</td>
<td>Ultra High</td>
</tr>
</tbody>
</table>

Security-related activities for requirements, coding, and testing nonexistent.

- **Basic**
  - Basic security-related activities for requirements, coding, and testing. Typical security functional features. Regular use of static analysis tools to detect security defects within the project.

- **Moderate**
  - Moderate security-related activities for requirements, design, coding, and testing. Additional security features (audit/log, cryptography). Identification and controlled update of third-party components' security patches. Routine use of static analysis and penetration testing tools. Security V&V activities conducted by an independent group.

- **Extensive**
  - Complex security requirements and threat modeling. Advanced secure-by-design security features. Extensive adversarial testing and security design/code review. Security assessment of third-party components and timely security patches updates. Thorough use of statics analysis, black-box, and penetration testing tools. V&V activities conducted by an independent group at the organization level.

- **Rigorous**
  - Extreme security requirements and threat modeling. Container-based approaches to advanced security features. Exhaustive adversarial testing, security design/code review, deep-dive analysis penetration testing, and use of formal methods throughout the lifecycle. Third-party components rigorously assessed and updated by a security science team. Maximal use of tools for static analysis, penetration testing, and black-box security testing. Use of formal verification and custom developed V&V tools. Security V&V activities conducted by an outside certified company.
Outline

• Background
• Previous studies on secure software development costs
• Proposed rating scale
  ➢ Data collection and Initial results of a Wide-band Delphi
• Next steps
Data Collection

Security experts estimates for the security parameter

Estimation experts estimates for the security parameter

Wideband Delphi

Industry

Projects’ Data ➔ Manual Data Collection Form

OSS

Projects’ Data ➔ Automated Data Collection

Projects’ Data ➔ Survey
Delphi Results

• Productivity Range (PR)
  • PR is the degree of influence a model parameter has on the estimated effort.
  • The Delphi session asked participants to rate the PR for security
  • The result is an average of 2.17 (table below)
  • A range of 2.17 means the parameter ratings from Nominal to Ultra High can change the effort estimate up to 117%, i.e., it costs 117% more to implement extreme security requirements and threat modeling

<table>
<thead>
<tr>
<th>Security Parameter Characteristic</th>
<th>Average (AVG)</th>
<th>Standard Deviation (SD)</th>
<th>Coefficient of Variation (SD/AVG)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security Requirements and Design</td>
<td>1.18</td>
<td>0.045</td>
<td>4%</td>
</tr>
<tr>
<td>Security Coding and Tools</td>
<td>1.18</td>
<td>0.130</td>
<td>11%</td>
</tr>
<tr>
<td>Security Verification &amp; Validation</td>
<td>1.56</td>
<td>0.152</td>
<td>10%</td>
</tr>
<tr>
<td><strong>Total Productivity Range</strong></td>
<td><strong>2.17</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Converting Delphi to a Rating Scale

- The Productivity Range is used to derive individual quantitative values, $EM_i$, for each rating level based on exponential growth
- Exponential growth is used in all COCOMO III parameters

**Exponential Growth**

\[ EM_i = (1 + r)^i \]

\[ r = \text{range}^{(1/n)} - 1 \]

- $EM_i$ is the effort multiplier for level $i$
- $i$ is the security level (Nominal is 0)
- $r$ is the growth rate
- range is the range of the effort multiplier
- $n$ is the index of the highest level ($i=4$)

**Security Rating Scale**

- Exponential growth is used in all COCOMO III parameters

![Security Rating Scale Diagram](image)
Outline

• Background
• Previous studies on secure software development costs
• Proposed rating scale
• Data collection and Initial results of a Wide-band Delphi

➢ Next steps
Next Steps

• Collect project and OSS data
• Calibrate the COCOMO III model
  • Functional size measures as well as traditional lines of code
  • Domain classification of software applications
  • All the model parameters
• Validate the security rating scale and its impact on effort
Thank you!

Elaine Venson  
venson@usc.edu

Brad Clark  
clarkbk@usc.edu

September 16, 2020
References


- Sammy Migues, John Steven, Mike Ware, 2019. Building Security in Maturity Model (BSIMM) - Version 10 (No. 10). Synopsys Software Integrity Group.

