Getting software security Right

Haiyun Xu, Theodoor Scholte
Table of contents

1. Who is SIG?

2. SIG software maintainability model

3. Getting software security Right: security by design

4. SIG ongoing security research

5. Potential research topics for student projects
SIG Background

- Spin-off from CWI in 2000, self-owned, independent
- Management advisory, fact-based
- Accredited software analysis lab employs analysis tools and models

SIG Service Examples

**Software Risk Assessment**
Reduce or eliminate technical and operational issues by identifying software system risks.

**Software Risk Monitor**
Monitor systems under development and maintenance to improve the processes involved in delivering high-quality software systems.

**Security Risk Assessment**
Gain control over IT security and prevent security incidents by identifying the root causes of weak spots in code, design and process.
Introduction myself

Dr. Haiyun Xu
Chief Security Officer / Researcher

Main security activities

• Research on Software Security Model
• Information Security Management
• Security analysis of Software Defined Networking
• Supervising master students on security research projects
Table of contents

1. Who is SIG?

2. SIG software maintainability model

3. Getting software security Right: security by design

4. SIG ongoing security research

5. Potential research topics for student projects
Introducing SIG maintainability model

The ISO 25010 standard for software quality

Software Quality
ISO 25010

- Usability
- Reliability
- Security
- Maintainability
- Portability
- Performance Efficiency
- Functional Suitability
- Compatibility
# Introduction to SIG maintainability model

The SIG/TÜViT quality model for software maintainability

## System properties

<table>
<thead>
<tr>
<th></th>
<th>Volume</th>
<th>Duplication</th>
<th>Unit size</th>
<th>Unit complexity</th>
<th>Unit interfacing</th>
<th>Module coupling</th>
<th>Component balance</th>
<th>Component independence</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Analysability</strong></td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td>×</td>
</tr>
<tr>
<td><strong>Modifiability</strong></td>
<td></td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Testability</strong></td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>×</td>
</tr>
<tr>
<td><strong>Modularity</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>×</td>
<td>× × ×</td>
</tr>
<tr>
<td><strong>Reusability</strong></td>
<td></td>
<td></td>
<td></td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Star ratings are relative to hundreds of systems in the SIG benchmark

- Ratings are between 1 and 5 stars, where 3 stars is market average maintainability

### Example

<table>
<thead>
<tr>
<th>RISK CATEGORY</th>
<th>PART OF SYSTEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low risk</td>
<td>65%</td>
</tr>
<tr>
<td>Moderate risk</td>
<td>15%</td>
</tr>
<tr>
<td>High risk</td>
<td>15%</td>
</tr>
<tr>
<td>Very high risk</td>
<td>5%</td>
</tr>
</tbody>
</table>
Table of contents

1. Who is SIG?

2. SIG software maintainability model

3. Getting software security Right: security by design

4. SIG ongoing security research

5. Potential research topics for student projects
Four different types of security verification

*They work best when applied together*

- Penetration tests
- Automated static code analysis
- Design/code reviews
- Build quality analysis

All security risks
Security by Design

Embedding software security in client organisation

Risk management and acceptance

Environmental factors
- Legal requirements, policies, constraints

Accountability
- Internal (risk dashboard), external (certification, compliance)

Risk analysis
- Inventory of threats, measures and risks

Threats, reqs, risks

- Design review
- Code review
- Security Test
- Pentest

Req’ments
- Training & Guidelines

Design
- QA Advice & Tuning

Build
- Accept

Test
- Deploy

Development phases
- SIG offer
Security by Design

SIG services

Step 1. How can I get secure software?
- Security governance maturity scan (based on Grip on SSD)
- Security requirements review

How can I make secure software?
- Security requirements review
- Secure design review
- Secure coding training & guidelines
- Secure development process review

How secure is my software?
- Security Risk Assessment
- Security compliance check (with whatever standard)
- Security inspection
- Security sample assessment (one use-case)

Threats, reqs, risks

Req’ments

Design review

Design

Code review

Build

Security Test

Test

Accept

Deploy

Training & Guidelines

QA Advice & Tuning

Pentest

Risk management steps
Development phases
SIG offer

Haiyun Xu, Theodoor Scholte, April 24 2015, Radboud University Nijmegen
SIG security inspection methodology

How SIG broke down security

System-level security

Confidentiality & Integrity

Non-repudiation & Accountability

Authenticity

Access Management Strength

Session Management Strength

Security User Management

Authentication Method Strength

Authentication Implement Strength

Authentication Enforcement

Haiyun Xu, Theodoor Scholte, April 24 2015, Radboud University Nijmegen
**SIG security inspection methodology**

The SIG ISO25010 Quality Model for security

---

### System properties

<table>
<thead>
<tr>
<th>Secure data transport</th>
<th>Identification strength</th>
<th>Access management strength</th>
<th>Session management strength</th>
<th>Authorized access</th>
<th>Input and output verification</th>
<th>Secure data storage</th>
<th>Evidence strength</th>
<th>Secure user management</th>
<th>Final result: ★★★★★☆</th>
</tr>
</thead>
</table>

- **Confidentiality & Integrity**: X
  - Final result: ★★★★★☆

- **Non-repudiation & Accountability**: X
  - Final result: ★★★★★☆

- **Authenticity**: X
  - Final result: ★★★★★☆

---

**ISO 25010 Security sub-characteristics**

The model is applied using a mix of tooling and expert review, from four perspectives.

---

Haiyun Xu, Theodoor Scholte, April 24 2015, Radboud University Nijmegen
### Example findings from a software security inspection

#### SIG security inspection methodology

1. **Finding**
   - 1. Missing certificate check by mobile app is abused with “man in the middle attack”
   - 2. Passwords leak because passwords are stored in database and can be decrypted
   - 3. IAM is hacked since URL and version are exposed
   - 4. Password leak from registration log
   - 5. Developers create data leak because of complexity in website architecture
   - 6. Backdoor in application interface is abused
   - 7. Injection attack abuses browser-only input for validation of data of birth
   - 8. Attack is not identified because logging/monitoring of key processes is not in place

#### Probability x Impact Matrix

<table>
<thead>
<tr>
<th>Probability</th>
<th>Impact</th>
<th>Finding</th>
</tr>
</thead>
<tbody>
<tr>
<td>low</td>
<td>low</td>
<td>8. Attack is not identified because logging/monitoring of key processes is not in place</td>
</tr>
<tr>
<td>low</td>
<td>med</td>
<td>7. Injection attack abuses browser-only input for validation of data of birth</td>
</tr>
<tr>
<td>low</td>
<td>high</td>
<td>6. Backdoor in application interface is abused</td>
</tr>
<tr>
<td>med</td>
<td>low</td>
<td>5. Developers create data leak because of complexity in website architecture</td>
</tr>
<tr>
<td>med</td>
<td>med</td>
<td>4. Password leak from registration log</td>
</tr>
<tr>
<td>med</td>
<td>high</td>
<td>3. IAM is hacked since URL and version are exposed</td>
</tr>
<tr>
<td>high</td>
<td>low</td>
<td>2. Passwords leak because passwords are stored in database and can be decrypted</td>
</tr>
<tr>
<td>high</td>
<td>med</td>
<td>1. Missing certificate check by mobile app is abused with “man in the middle attack”</td>
</tr>
</tbody>
</table>

#### Effort Indicator

- Diameter of circle represents indication of effort
- 3 manmonths
- (3+) 5 manmonths

---

Haiyun Xu, Theodoor Scholte, April 24 2015, Radboud University Nijmegen
Table of contents

1. Who is SIG?

2. SIG software maintainability model

3. Getting software security Right: *security by design*

4. SIG ongoing security research

5. Potential research topics for student projects
Ongoing Security Research

**Metric-based Security Model**

System's Source Code

Haiyun Xu, Theodoor Scholte, April 24 2015, Radboud University Nijmegen
Ongoing Security Research

Automated Security Web Scanners

Why are the tools not used in development process?

• Many vulnerability scanners exist yet they are rarely being used!
• What are important features needed by companies?
• Do the tools provide these features?
• What needs to improved to make tools usable?
Ongoing Security Research

Security Analysis of Software Defined Networking (SDN)

- Apply STRIDE and Attack tree for threat modeling
- Design test cases and perform using SDN tools
- ...

- Security assessment approach

Threat Modeling

Assessment

SDN Security Research

Mitigation

Testing

- Mitigation solutions and best practices
- Security application
Table of contents

1. Who is SIG?

2. SIG software maintainability model

3. Getting software security Right: security by design

4. SIG ongoing security research

5. Potential research topics for student projects
Security Code Review

- How to perform security code review effectively / efficiently?
  - Are the security standards / guidelines be helpful (e.g., OWASP ASVS)?
  - Are the penetration testing tools helpful?
  - Are the static analysis tools helpful (e.g., Fortify)?
- How to design the code reviewer strategy?
  - Requirements on code review expertise on security?
  - Will it be helpful if we group code reviewers by skill level?
Potential Topic For Student Projects

Patching and Security

User aspect
- Does patching faster improve security?
- Maintainability of library, known vulnerabilities

Vendor aspect
- Do we want a policy that skips patching in favor of rolling software fast enough to make it a moving target?
- When patching, is that better to disclose or keep the fix secret?

Design a cost model for patching
Potential Topic For Student Projects

More open projects online

Open projects:

- http://www.cs.ru.nl/J.Visser/students/open-projects/

Open projects

Are you a student looking for a research project? If any of the following projects:

- Getting data right: separate knowledge from noise in distributed datasets
- Cyber security in Europe: combine multiple sources
- Making security measurable: develop new KPIs that bridge software security and business processes
- Modelling enterprise IT: build and test a model of enterprise IT projects, and suppliers
- Your personal software quality timeline: build and field test an app that provides software engineers with a targeted, personalised event stream about the product they are working on and the team they are part of
- Managing value at risk: develop a monitoring model for (agile) software development that tracks the probability that high-value features will actually be delivered and will actually start creating value
Haiyun Xu

+31 6 2312 3519

h.xu@sig.eu