Agile Secure Software Development in a Large Software Development Organisation

Security Testing

Achim D. Brucker
achim.brucker@sap.com http://www.brucker.ch/
SAP SE, Vincenz-Priessnitz-Str. 1, 76131 Karlsruhe, Germany

ASSD Keynote
First International Workshop on Agile Secure Software Development (ASSD)
Toulouse, France, August 24–28, 2015
Abstract

Security testing is an important part of any (agile) secure software development lifecycle. Still, security testing is often understood as an activity done by security testers in the time between “end of development” and “offering the product to customers.”

Learning from traditional testing that the fixing of bugs is the more costly the later it is done in development, we believe that security testing should be integrated into the daily development activities. To achieve this, we developed a security testing strategy, as part of SAP’s security development lifecycle which supports the specific needs of the various software development models at SAP.

In this presentation, we will briefly presents SAP’s approach to an agile secure software development process in general and, in particular, present SAP’s Security Testing Strategy that enables developers to find security vulnerabilities early by applying a variety of different security testing methods and tools.
Agenda

1. Background
2. Motivation
3. Risk-based Security Testing as Part of SAP’s $S^2$DL
4. Lesson’s Learned
5. How Does This Resonate With Agile Development?
• Leader in Business Software
  • Cloud
  • Mobile
  • On premise

• Many different technologies and platforms, e.g.,
  • In-memory database and application server (HANA)
  • Netweaver for ABAP and Java

• More than 25 industries

• 63% of the world’s transaction revenue touches an SAP system

• over 68,000 employees worldwide
  over 25,000 software developers

• Headquarters: Walldorf, Germany (close to Heidelberg)
Personal Background

• I wear two hats:
  • (Global) Security Testing Strategist
  • Research Expert/Architect

Working for the central software security team

• Background:
  Security, Formal Methods, Software Engineering

• Current work areas:
  • Static code analysis
  • (Dynamic) Security Testing
  • Mobile Security
  • Security Development Lifecycle
  • Secure Software Development Lifecycle

http://www.brucker.ch/
SAP Uses a De-centralised Secure Development Approach

• **Central security expert team (S²DL owner)**
  - Organizes security trainings
  - Defines product standard “Security”
  - Defines risk and threat assessment methods
  - Defines security testing strategy
  - Selects and provides security testing tools
  - Validates products
  - Defines and executes response process

• **Local security experts**
  - Embedded into development teams
  - Organize local security activities
  - Support developers and architects
  - Support product owners (responsibles)

• **Development teams**
  - Select technologies
  - Select development model
  - Design and execute security testing plan
  - …
Agenda

1. Background
2. Motivation
3. Risk-based Security Testing as Part of SAP’s S²DL
4. Lesson’s Learned
5. How Does This Resonate With Agile Development?
When Do We Fix Bugs

Source: Applied Software Measurement, Capers Jones, 1996
Microsoft’s SDL

Training
- Core Security Training
  - Establish Security Requirements
  - Create Quality Gates / Bug Bars
  - Security & Privacy Risk Assessment

Requirements
- Establish Design Requirements
- Analyze Attack Surface
- Threat Modeling

Design
- Use Approved Tools
- Deprecate Unsafe Functions
- Static Analysis

Implementation
- Dynamic Analysis
- Fuzz Testing
- Attack Surface Review

Verification
- Incident Response Plan
- Final Security Review
- Release Archive

Release
- Execute Incident Response Plan
Agenda

1 Background

2 Motivation

3 Risk-based Security Testing as Part of SAP’s S²DL

4 Lesson’s Learned

5 How Does This Resonate With Agile Development?
Our Start: SAST as a Baseline

SAST tools used at SAP:

<table>
<thead>
<tr>
<th>Language</th>
<th>Tool</th>
<th>Vendor</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABAP</td>
<td>CVA (SLIN_SEC)</td>
<td>SAP</td>
</tr>
<tr>
<td>JavaScript</td>
<td>Checkmarx CxSAST</td>
<td>Checkmarx</td>
</tr>
<tr>
<td>C/C++</td>
<td>Coverity</td>
<td>Coverity</td>
</tr>
<tr>
<td>Others</td>
<td>Fortify</td>
<td>HP</td>
</tr>
</tbody>
</table>

- Since 2010, mandatory for all SAP products
- Multiple billions lines analyzed
- Constant improvement of tool configuration
- Further details:
Combining Multiple Security Testing Methods and Tools

- Risks of only using only SAST
  - Wasting effort that could be used more wisely elsewhere
  - Shipping insecure software
- Examples of SAST limitations
  - Not all programming languages supported
  - Covers not all layers of the software stack
Combining Multiple Security Testing Methods and Tools

- Risks of only using only SAST
  - Wasting effort that could be used more wisely elsewhere
  - Shipping insecure software
- Examples of SAST limitations
  - Not all programming languages supported
  - Covers not all layers of the software stack
Combining Multiple Security Testing Methods and Tools

- Risks of only using only SAST
  - Wasting effort that could be used more wisely elsewhere
  - Shipping insecure software
- Examples of SAST limitations
  - Not all programming languages supported
  - Covers not all layers of the software stack
Combining Multiple Security Testing Methods and Tools

- Risks of only using only SAST
  - Wasting effort that could be used more wisely elsewhere
  - Shipping insecure software
- Examples of SAST limitations
  - Not all programming languages supported
  - Covers not all layers of the software stack
A Risk-based Test Plan

RISK ASSESSMENT
(e.g., SECURIM, Threat Modelling, OWASP ASVS)

• Combines multiple security testing methods, e.g., code scans, dynamic analysis, manual penetration testing or fuzzing
• Selects the most efficient test tools and test cases based on the risks and the technologies used in the project
• Re-adjusts priorities of test cases based on identified risks for the project
• Monitors false negative findings in the results of risk assessment
SAP’ Secure Software Development Lifecycle (S²DL)

Figure: SAP SSDL
Security Validation

- Acts as first customer
- Is not a replacement for security testing during development
- Security Validation
  - Check for “flaws” in the implementation of the S\(^2\)DL
  - Ideally, security validation finds:
    - No issues that can be fixed/detected earlier
    - Only issues that cannot be detect earlier
      (e.g., insecure default configurations, missing security documentation)
Security Validation

- Acts as first customer
- Is not a replacement for security testing during development
- Security Validation
  - Check for “flaws” in the implementation of the S²DL
  - Ideally, security validation finds:
    - No issues that can be fixed/detected earlier
    - Only issues that cannot be detect earlier
      (e.g., insecure default configurations, missing security documentation)

Penetration tests in productive environments are different:
- They test the actual configuration
- They test the productive environment (e.g., cloud/hosting)
How to Measure Success

• Analyze the vulnerabilities reported by
  • Security Validation
  • External security researchers

• Vulnerability not detected by our security testing tools
  • Improve tool configuration
  • Introduce new tools

• Vulnerability detected by our security testing tools
  • Vulnerability in older software release
  • Analyze reason for missing vulnerability
How to Measure Success

- Analyze the vulnerabilities reported by
  - Security Validation
  - External security researchers
- Vulnerability not detected by our security testing tools
  - Improve tool configuration
  - Introduce new tools
- Vulnerability detected by our security testing tools
  - Vulnerability in older software release
  - Analyze reason for missing vulnerability

Success criteria:
Percentage of vulnerabilities not covered by our security testing tools increases
How to Measure Success

- Analyze the vulnerabilities reported by
  - Security Validation
  - External security researchers
- Vulnerability not detected by our security testing tools
  - Improve tool configuration
  - Introduce new tools
- Vulnerability detected by our security testing tools
  - Vulnerability in older software release
  - Analyze reason for missing vulnerability

Success criteria:
Percentage of vulnerabilities not covered by our security testing tools increases
How to Measure Success

- Analyze the vulnerabilities reported by
  - Security Validation
  - External security researchers
- Vulnerability not detected by our security testing tools
  - Improve tool configuration
  - Introduce new tools
- Vulnerability detected by our security testing tools
  - Vulnerability in older software release
  - Analyze reason for missing vulnerability

**Success criteria:**
Percentage of vulnerabilities not covered by our security testing tools increases
Agenda

1. Background
2. Motivation
3. Risk-based Security Testing as Part of SAP’s S²DL
4. Lesson’s Learned
5. How Does This Resonate With Agile Development?
Key Success Factors

• A holistic security awareness program for
  • Developers
  • Managers

Yes, security awareness is important but Developer awareness is even more important!
Key Success Factors

- A holistic security awareness program for
  - Developers
  - Managers
- Yes, security awareness is important
Key Success Factors

- A holistic security awareness program for
  - Developers
  - Managers

- Yes, security awareness is important **but**
Key Success Factors

- A holistic security awareness program for
  - Developers
  - Managers

- Yes, security awareness is important **but**

  Developer awareness is even more important!
Listen to Your Developers!

We are often talking about a lack of security awareness and, by that, forget the problem of lacking development awareness.

- Building a secure system more difficult than finding a successful attack.
- Do not expect your developers to become penetration testers (or security experts)!
Security testing tools for developers, need to:

- Be applicable from the start of development
- Automate the security knowledge
- Be deeply integrated into the dev. env., e.g.,
  - IDE (instant feedback)
  - Continuous integration
- Provide easy to understand fix recommendations
- Declare their “sweet spots”
Collaborate!

Security experts need to collaborate with development experts to

• Create easy to use security APIs (ever tried to use an SSL API securely)
• Create languages and frameworks that make it hard to implement insecure systems
• Explain how to program securely
Agenda

1. Background
2. Motivation
3. Risk-based Security Testing as Part of SAP’s $^{2}$DL
4. Lesson’s Learned
5. How Does This Resonate With Agile Development?
Agile Development

- What is agile for you?
  SCRUM, Continuous Delivery, DevOps, SCRUM, Cloud development, . . .
- Cloud/agile development lifecycle
Secure Agile Development

- Level of Trust
- Risk Identification
- Threat Modelling
- Security Measures
- Security Testing
- PSC Security

- Risk Mitigation & Testing
- Secure Programming
- Static Testing
- Dynamic Testing
- Manual Testing
- Security Validation

- Design Thinking
- Define Strategic Planning
- Build Agile Software Engineering
- Operate Understand Customers
- Release Early Access
- Security Response

© 2015 SAP SE. All Rights Reserved.
Open (Research) Questions

- **Social aspects**
  - Does the SecDevOps model increase security awareness? (Developers and their managers are also responsible for operational risks)
  - Does this impact the willingness to take (security) risks and/or the risk assessment?

- **Process and organisational aspects**
  - What services should be offered centrally?
  - How to ensure a certain level of security across all products?
  - How to ensure a certain level of security across the end-to-end supply chain?

- **Technical and fundamental aspects**
  - How do we need to adapt development support
  - How do we need to adapt threat modelling or risk assessment methods
  - How do we need to adapt security testing techniques

- **The big challenge in practice:**
  
  Products are often offered in the cloud (SaaS) and on premise
Thank you!

http://xkcd.com/327/

Dr. Achim D. Brucker
achim.brucker@sap.com
http://www.brucker.ch/
Related Publications

Ruediger Bachmann and Achim D. Brucker.
Developing secure software: A holistic approach to security testing.

Achim D. Brucker, Lukas Brügger, and Burkhart Wolff.
Formal firewall conformance testing: An application of test and proof techniques.

Achim D. Brucker and Uwe Sodan.
Deploying static application security testing on a large scale.
ISBN 978-3-88579-622-0.

Achim D. Brucker and Burkhart Wolff.
On theorem prover-based testing.
ISSN 0934-5043.