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 present 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.

© 2015 SAP SE. All Rights Reserved.
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/

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?

Vulnerability Distribution

- 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

- Development teams
  - Select technologies
  - Select development model
  - Design and execute security testing plan
  - ...

- 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
  - ...

- Local security experts
  - Embedded into development teams
  - Organize local security activities
  - Support developers and architects
  - Support product owners (responsibles)
When Do We Fix Bugs

Microsoft’s SDL

Agenda

1 Background
2 Motivation
3 Risk-based Security Testing as Part of SAP’s S^2DL
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

- **Client Application**
- **Web Browser**
- **Server Application**
- **Runtime Container**
- **Backend Systems**

<table>
<thead>
<tr>
<th>Checkmarx (JavaScript)</th>
<th>Web Browser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fortify (Java)</td>
<td>Runtime Container</td>
</tr>
<tr>
<td>DOMinator</td>
<td>Backend Systems</td>
</tr>
</tbody>
</table>

- **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

© 2015 SAP SE. All Rights Reserved. Page 13 of 28
A Risk-based Test Plan

- 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

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

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

© 2015 SAP SE. All Rights Reserved. Page 17 of 28
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

© 2015 SAP SE. All Rights Reserved. Page 18 of 28
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 for Developers

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 S²DL
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

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?
- Technical and fundamental aspects
  - How do we need to adapt development support?
  - How do we need to adapt threat modelling or risk assessment methods?
- The big challenge in practice:
  - Products are often offered in the cloud (SaaS) and on premise