Bringing Security Testing to Development

How to Enable Developers to Act as Security Experts
Background: SAP SE

• SAP SE
  – Business Software Vendor
  – Over 68000 employees
  – Worldwide development

• Myself
  – Security Testing Strategist
  – Researcher
  – Working in the central Software Security Team
De-centralized Secure Development Model

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 dev. teams
- Organize local security activities
- Support developers and architects
- Support product owners/responsibles

Development Teams
- Select technologies
- Select development model
- ...
MOTIVATION
Vulnerability Distribution

Source: www.cvedetails.com
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

- Design
  - Analyze Attack Surface
  - Threat Modeling

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

- Verification
  - Dynamic Analysis
    - Fuzz Testing
    - Attack Surface Review

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

- Response
  - Execute Incident Response Plan
RISK BASED SECURITY TESTING AS PART OF SAP’S S²DL
Our Start: SAST as Baseline

- Mandatory since 2010 for all products
- Multiple billons lines analyzed
- Constant improvements:
  - tool configuration (e.g., based on feedback from development, validation, response)
  - new tools and methods

### Language Tool Vendor

<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>C/C++</td>
<td>Coverity</td>
<td>Coverity</td>
</tr>
<tr>
<td>JavaScript, Ruby</td>
<td>Checkmarx</td>
<td>Checkmarx</td>
</tr>
<tr>
<td>Others</td>
<td>Fortify</td>
<td>HP</td>
</tr>
</tbody>
</table>
Are We Done?

- **SAST Advantages**
  - Early in Development
  - Wide range of vuln. Types
  - Good fix instructions

- **SAST Limitations**
  - Quality depends on programming language used
  - Usually covers only one layer of the application stack
Are We Done?

• SAST Advantages
  – Early in Development
  – Wide range of vuln. Types
  – Good fix instructions

• SAST Limitations
  – Quality depends on programming language used
  – Usually covers only one layer of the application stack
Are We Done?

- **SAST Advantages**
  - Early in Development
  - Wide range of vulnerability types
  - Good fix instructions

- **SAST Limitations**
  - Quality depends on programming language used
  - Usually covers only one layer of the application stack
Are We Done?

• **SAST Advantages**
  - Early in Development
  - Wide range of vuln. Types
  - Good fix instructions

• **SAST Limitations**
  - Quality depends on programming language used
  - Usually covers only one layer of the application stack

![Diagram showing software stack with tools like Checkmarx, DOMinator, Fortify (Java), HP WebInspect]
How To Select The Best Tools

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

Prioritized SAP Security Req.
Implement. Details
Application Type

Security Test Plan
Example: Security Test Plan

- **Mobile Device**
  - Risk: Attacker might inject JavaScript (XSS)
  - Security Control 1: Use only UI5 controls
  - Assumption: SAP Kapsel with SMP and Afaria
    - Test: Static Code Analysis using Checkmarx
      - Justification: recommended tool
      - Expected Coverage: all client-side JavaScript code
      - Expected Effort: 10min per development day (ramp-up not included)
  - Security Control 2: use only SSL connections with valid certificates
    - Test 1: Static Code Analysis for finding non-https connections
      - Justification: low effort, already included in test for Security Control 1
      - Expected Coverage: all client-side JavaScript code
      - Expected Effort: included in effort for scans for Security Control 1
    - Test 2: Manual test with invalid certs (e.g., self-signed, own CA)
      - Justification: no automated tool available, self-signed certificates allowed during development
      - Expected Coverage: all https connections used for accessing the Web Server
      - Expected Effort: ½ day towards the end of development

- **Web Server / Web Application (...)**
Example: Security Test Report

• **Mobile Device**
  - Risk: Attacker might inject JavaScript (XSS)
  - Security Control 1: Use only UI5 controls
  - Assumption: SAP Kapsel with SMP and Afaria
    - Test: Static Code Analysis using Checkmarx
      » Result: no issues
      » Actual Coverage: all client-side JavaScript code
      » Actual Effort: total effort 2 days (15min per day, instead of expected 10)
  - Security Control 2: use only SSL connections with valid certificates
    - Test 1: Static Code Analysis for finding non-https connections
      » Result: exempted one issue
      » Actual Coverage: all client-side JavaScript code
      » Actual Effort: included in effort for scans for Security Control 1
    - Test 2: Manual test with invalid certs (e.g., self-signed, own CA)
      » Expected Coverage: all https connections used for accessing the Web Server
      » Expected Effort: ½ day towards the end of development

• **Web Server / Web Application (...)**
SAP’s S²DL

Start of development

Preparation
- Training
  - Security awareness
  - Secure programming
  - Threat modelling
  - Security static analysis
  - Data protection and privacy
  - Security expert curriculum
- Risk Identification
  - SECURIM (Security Risk Identification and Management)
  - Data Privacy Impact Assessment
  - Threat Modeling
- Plan Security Measures
  - Plan product standard compliance
  - Plan security features
  - Plan security tests
  - Plan security response
- Secure development
  - Secure programming
  - Static code scan
  - Code review
- Security testing
  - Dynamic testing
  - Manual testing
  - External security assessment

Transition

Utilization

Release decision

- Security Response
- Independent security assessment
- Execute the security response plan
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 $S^2DL$
- 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)

- Note, 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
LESSONS LEARNED
Key Success Factor

• A holistic security awareness program for
  – Developers
  – Managers
Key Success Factor

• A holistic security awareness program for
  – Developers
  – Managers
• Yes, security awareness is important - but
Key Success Factor

- 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
Conclusion

• **Secure software development is a**
  - Prerequisite for the secure and compliant operation: We need SecDevOps!
  - Risk of operating and maintaining IT systems

• **Security requires an end-to-end approach**
  - Training of developers, architects, product owners
  - Security testing during development
  - Validation of your security testing efforts
  - Maintenance and security patch management

• **Developers are your most important ally**
  - Make life easy for them
Thank You

Contact Details:

• Achim D. Brucker
  www.brucker.ch
  achim.brucker@sap.com

• Stephen Hookings
  stephen.hookings@sap.com

• Dimitar Yanev
  dimitar.yanev@sap.com
Bibliography