

Bringing Security Testing to Development

*How to Enable Developers to
Act as Security Experts*

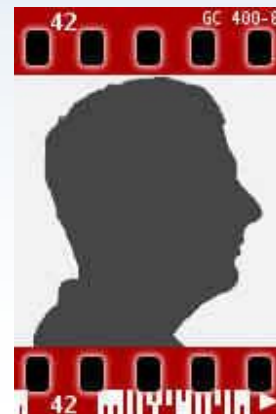


OWASP AppSecEU 15
Amsterdam, The Netherlands

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
- ...

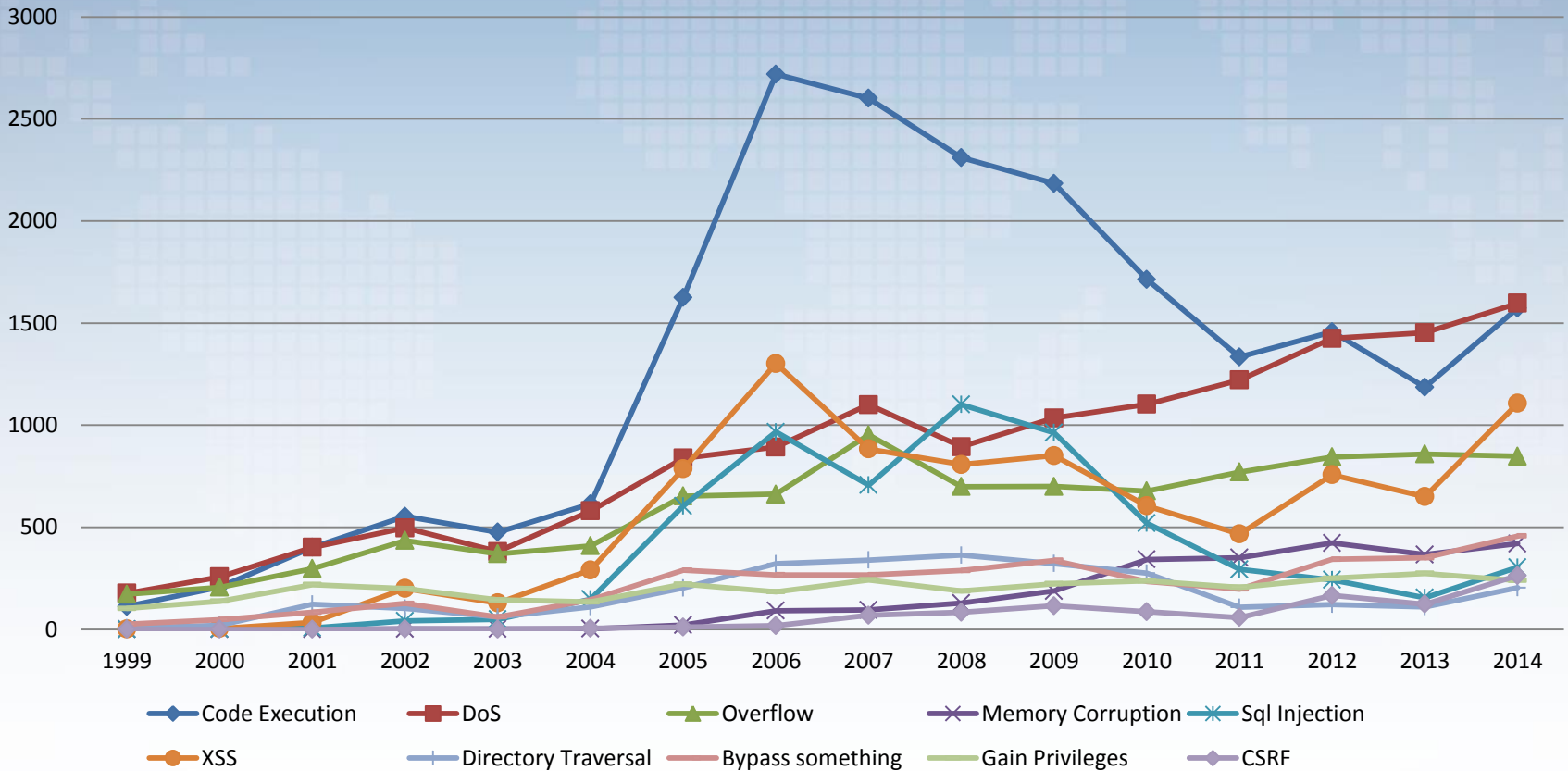


MOTIVATION



OWASP AppSecEU 15
Amsterdam, The Netherlands

Vulnerability Distribution

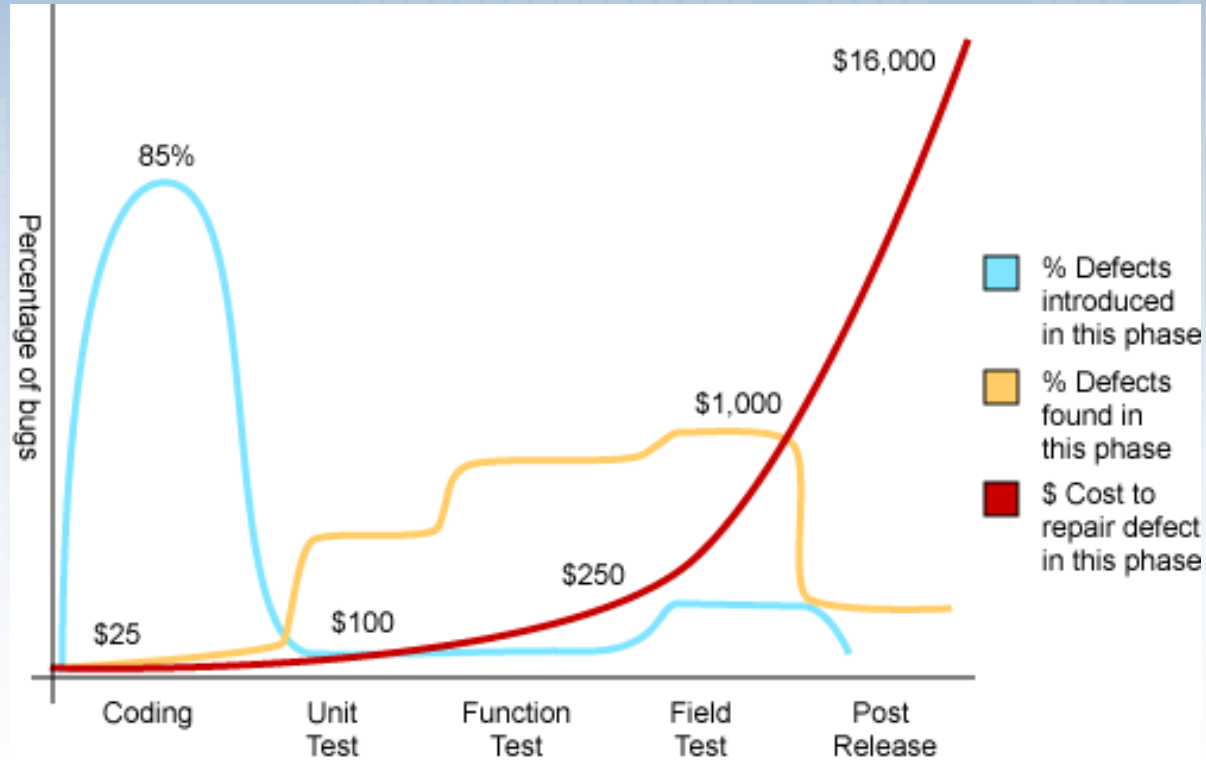


Source: www.cvedetails.com



OWASP AppSecEU 15
Amsterdam, The Netherlands

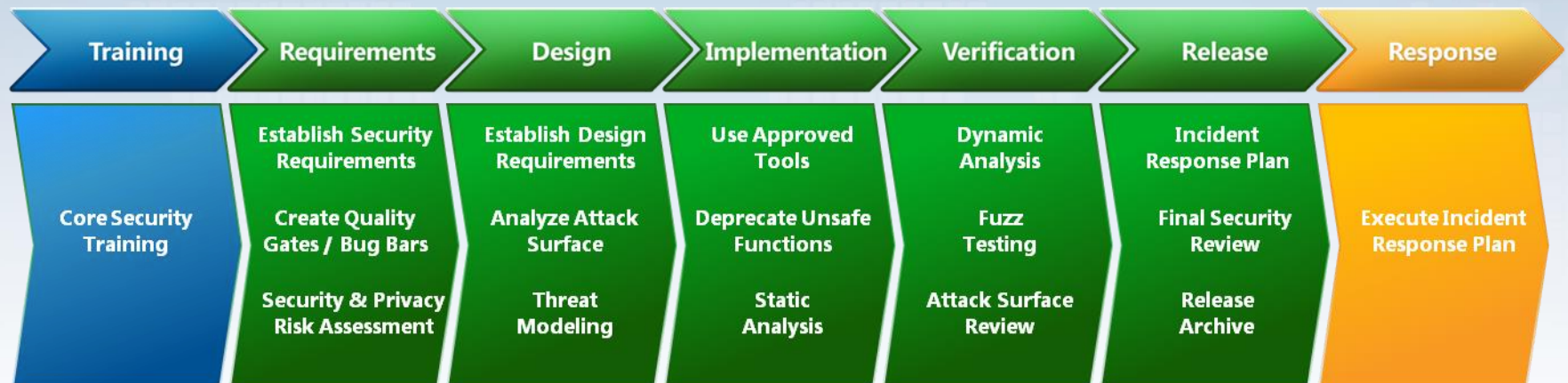
When Do We Fix Bugs?



Source: Applied Software Measurement, Capers Jones, 1996



Microsoft's SDL

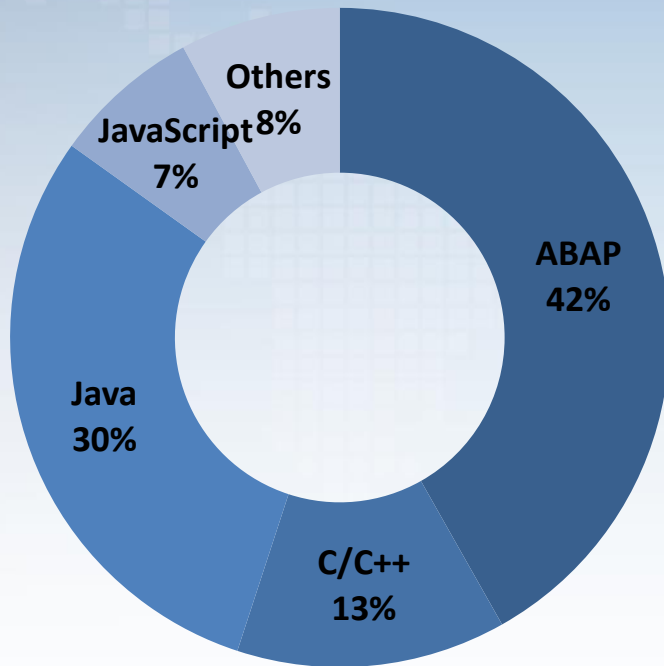


RISK BASED SECURITY TESTING AS PART OF SAP'S S²DL



OWASP AppSecEU 15
Amsterdam, The Netherlands

Our Start: SAST as Baseline



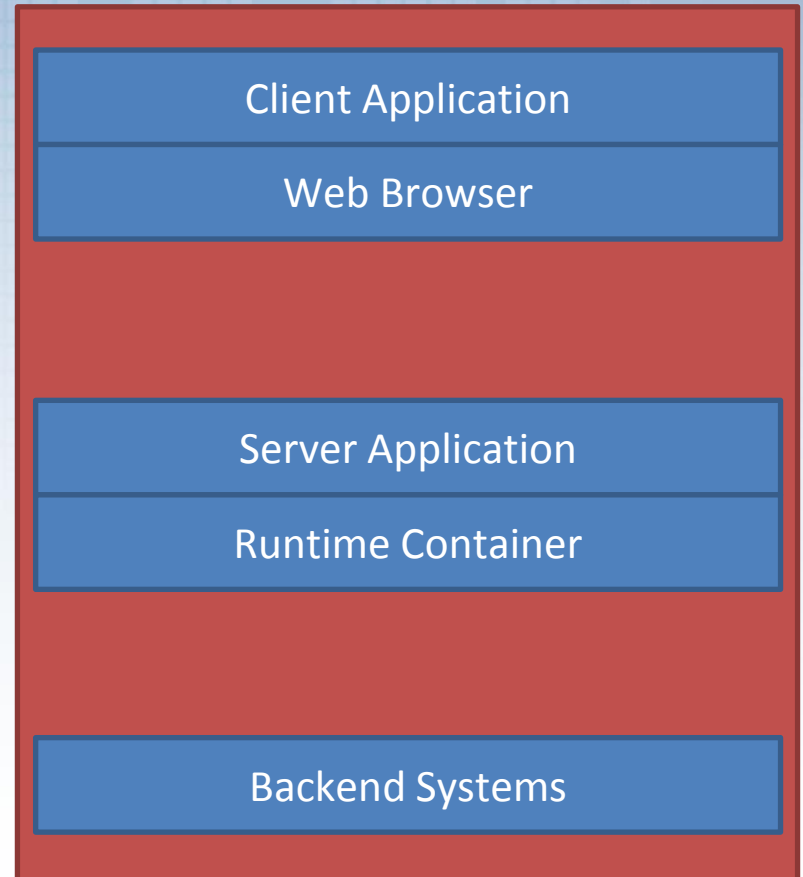
Language	Tool	Vendor
ABAP	CVA (SLIN_SEC)	SAP
C/C++	Coverity	Coverity
JavaScript, Ruby	Checkmarx	Checkmarx
Others	Fortify	HP

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



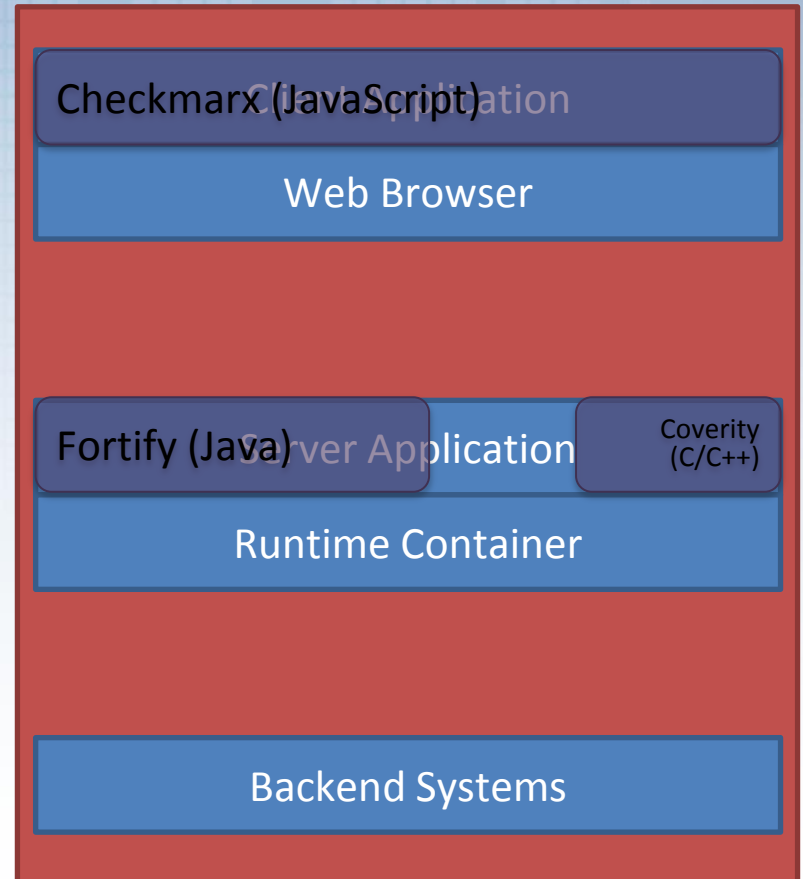
Are We Done?

- **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



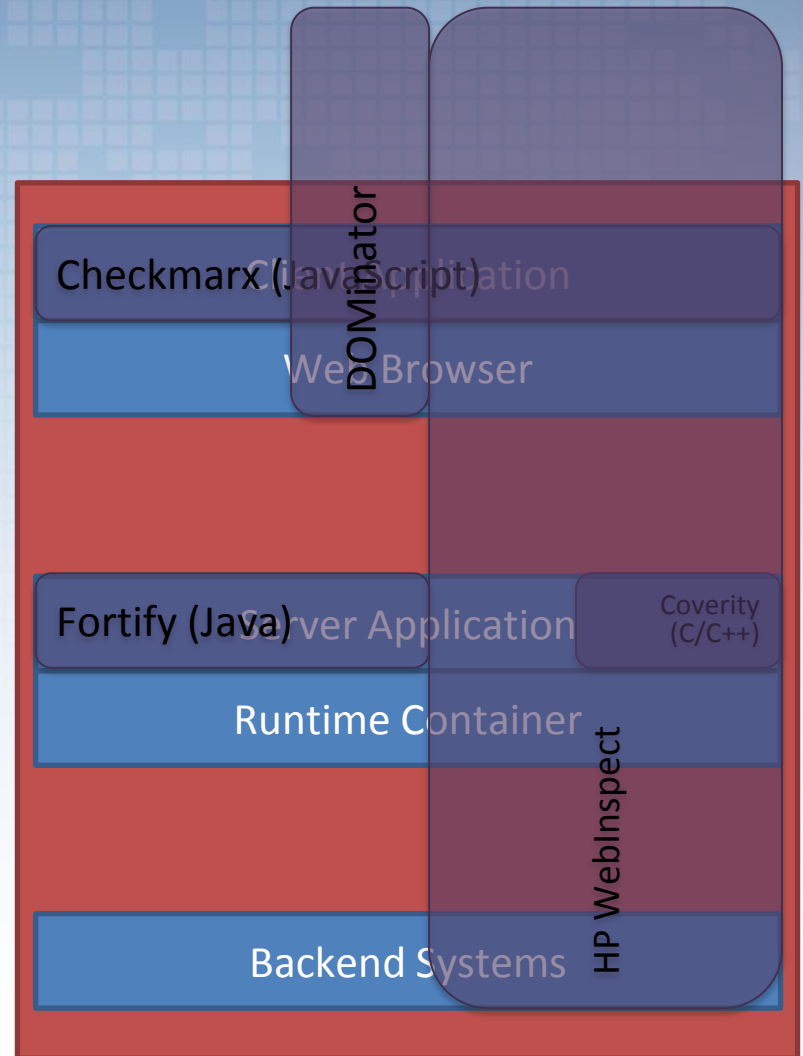
Are We Done?

- **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



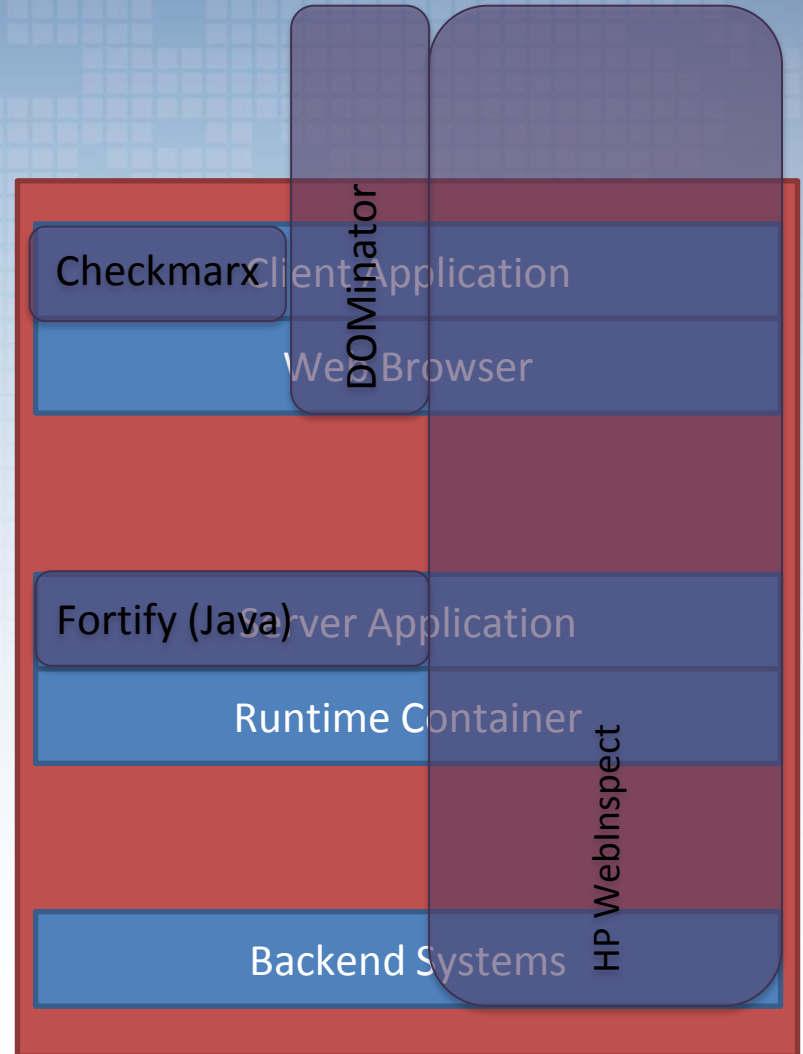
Are We Done?

- **SAST Advantages**
 - Early in Development
 - Wide range of 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?

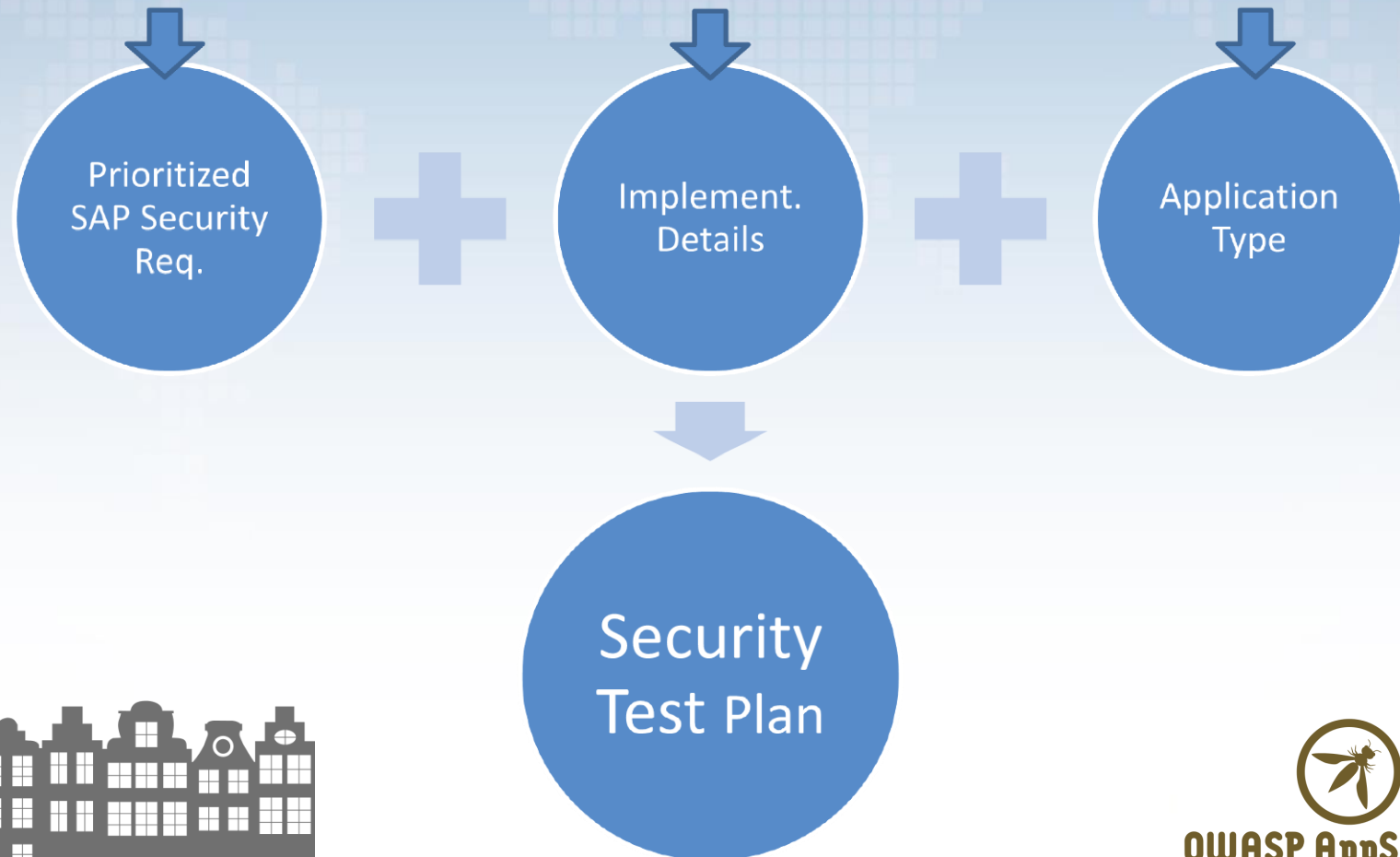
- **SAST Advantages**
 - Early in Development
 - Wide range of vuln. Types
 - Good fix instructions
- **SAST Limitations**
 - Quality depends on programming language used
 - Usually covers only one layer of the application stack



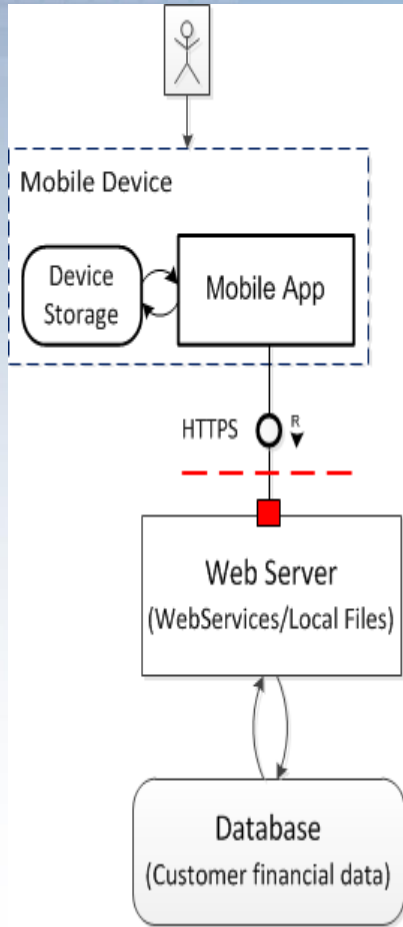
How To Select The Best Tools

RISK ASSESMENT

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



Example: Security Test Plan



• Mobile Device

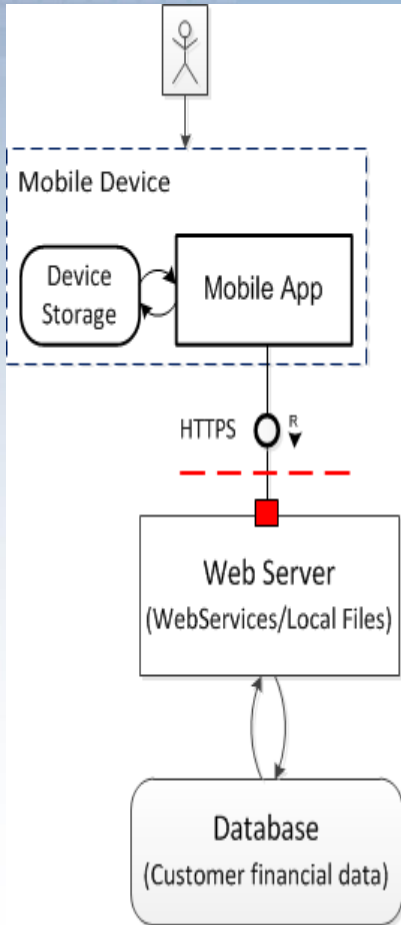
- Risk: Attacker might inject JavaScript (XSS)
- Security Control 1: Use only UI5 controls
- Assumption: SAP Kapsel with SMP and Afaria
 - Test: Static Code Analysis using Checkmarx
 - » Justification: recommended tool
 - » Expected Coverage: all client-side JavaScript code
 - » Expected Effort: 10min 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 (...)

Illustrative
Example



Example: Security Test Report

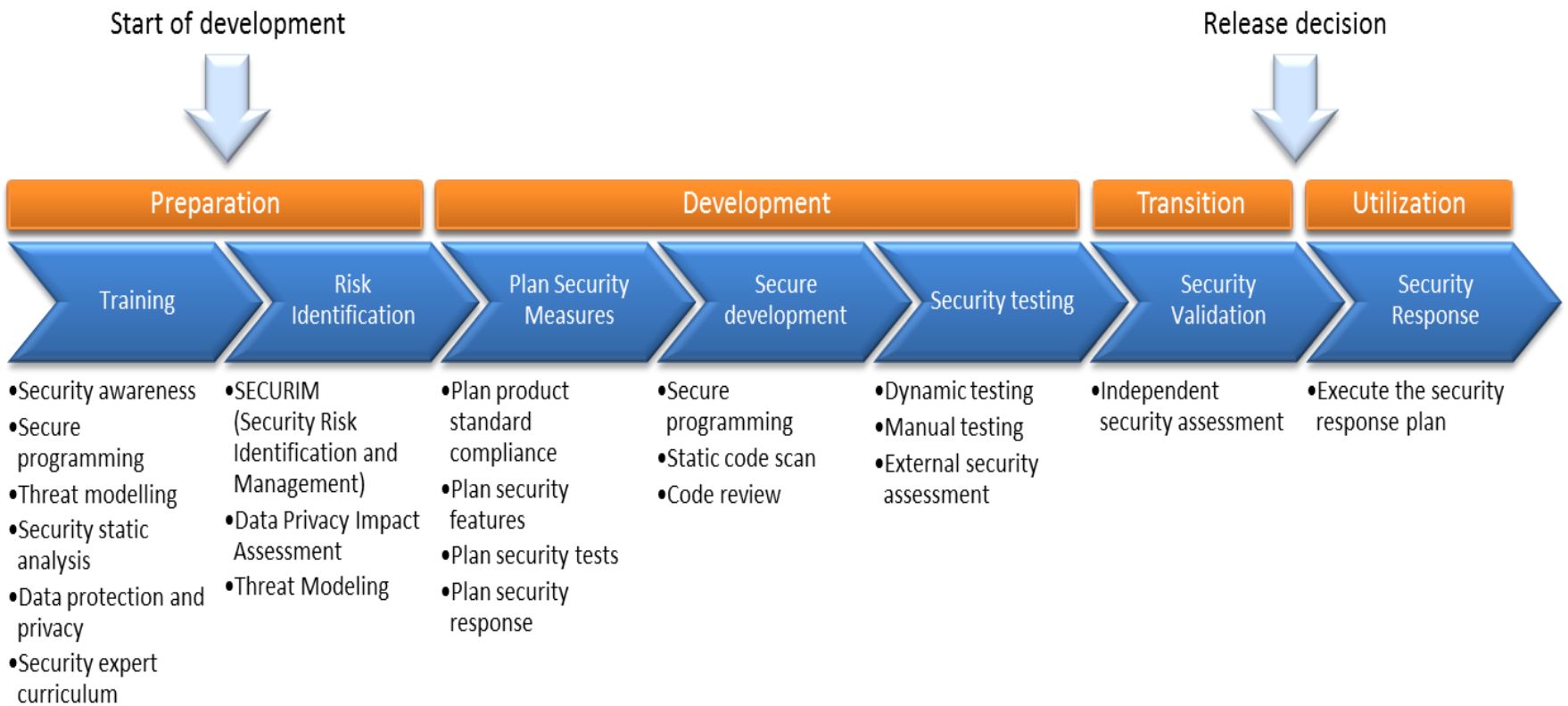


- **Mobile Device**
- Risk: Attacker might inject JavaScript (XSS)
- Security Control 1: Use only UI5 controls
- Assumption: SAP Kapsel with SMP and Afaria
 - Test: Static Code Analysis using Checkmarx
 - » Result: no issues
 - » Actual Coverage: all client-side JavaScript code
 - » Actual Effort: total 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 (...)**

Illustrative
Example



SAP's S²DL



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)
- 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



OWASP AppSecEU 15
Amsterdam, The Netherlands

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



OWASP AppSecEU 15
Amsterdam, The Netherlands

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

- <http://www.sap.com/security>
- Ruediger Bachmann and Achim D. Brucker. Developing secure software: A holistic approach to security testing. *Datenschutz und Datensicherheit (DuD)*, 38(4):257–261, April 2014.
<http://www.brucker.ch/bibliography/abstract/bachmann.ea-security-testing-2014>
- Achim D. Brucker and Uwe Sodan. Deploying static application security testing on a large scale. In Stefan Katzenbeisser, Volkmar Lotz, and Edgar Weippl, editors, *GI Sicherheit 2014*, volume 228 of *Lecture Notes in Informatics*, pages 91–101. GI, March 2014.
<http://www.brucker.ch/bibliography/abstract/brucker.ea-sast-experiences-2014>

