

# Security Testing: Myths, Challenges, and Opportunities

## Experiences in Integrating Security Testing “End-to-End” Into the Software Life-Cycle at SAP

**Achim D. Brucker**

achim.brucker@sap.com

<http://www.brucker.ch/>

SAP SE, Vincenz-Priessnitz-Str. 1, 76131 Karlsruhe, Germany

**SECTEST Keynote**

6th international Workshop on Security Testing (SECTEST)

Graz, Austria, April 13, 2015

# ***Security Testing: Myths, Challenges, and Opportunities***

Experiences in Integrating Security Testing “End-to-End” Into the Software Life-Cycle at SAP

## Abstract

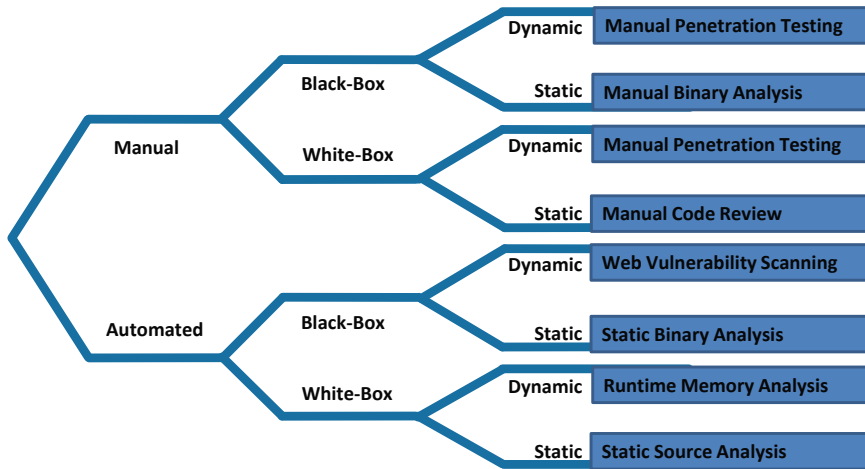
Security testing is an important part of any security development lifecycle (SDL) and, thus, should be a part of any software (development) lifecycle. Still, security testing is often understood as an activity done by security testers in the time between “end of development” and “offering the product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

On the one hand, learning from traditional testing that the fixing of bugs is the more costly the later it is done in development, security testing should be integrated into the daily development activities. On the other hand, developing software for the cloud and offering software in the cloud raises the need for security testing in a “close-to-production” or even production environment. Consequently, we need an end-to-end integration of security testing into the software lifecycle.

In this talk, we will report on our experiences on integrating security testing “end-to-end” into SAP’s software development lifecycle in general and, in particular, SAP’s Secure Software Development Lifecycle (S<sup>2</sup>DL).

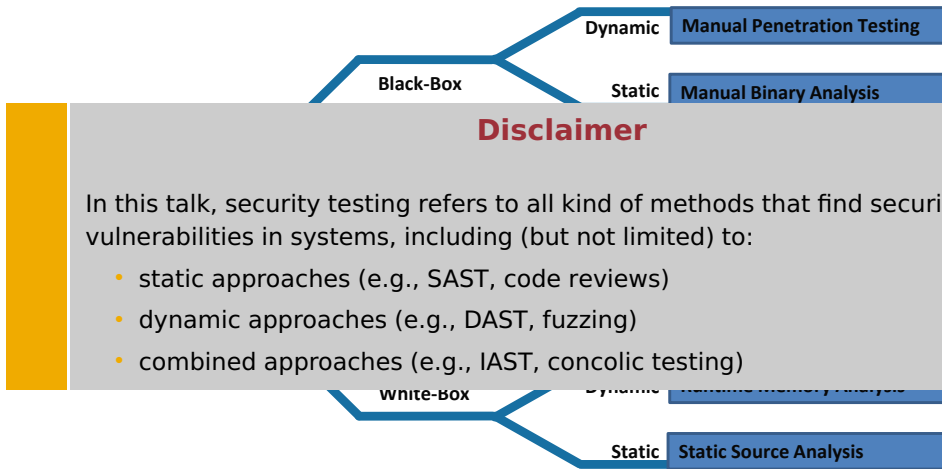
Moreover, we will discuss different myths, challenges, and opportunities in the area security testing.

# A Security Testing Taxonomy



# A Security Testing Taxonomy

... and a Disclaimer



# Agenda

---

- 1 SAP in a Nutshell
- 2 Motivation
- 3 The Beginning: Large Scale Introduction of SAST
- 4 A Risk-based Security Testing Strategy
- 5 SAP's Secure Software Development Lifecycle (S<sup>2</sup>DL)
- 6 Myths and Lesson's Learned

# Die SAP SE

- 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
- approx. 68 000 employees worldwide
- Headquarters: Walldorf  
(close to Heidelberg, Germany)

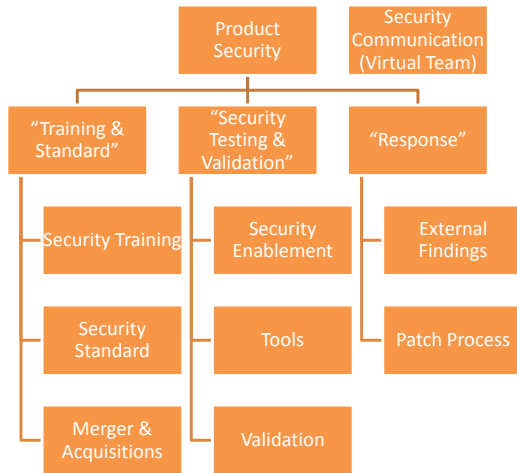


# SAP' Security Team

## How SAP Organizes Software Security

### De-centralized development model:

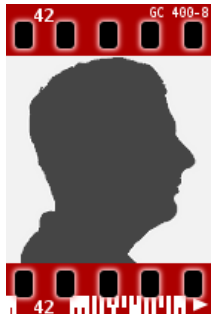
- **Central security expert team** (S<sup>2</sup>DL owner)
  - Organizes security trainings
  - Defines product standard "Security"
  - Defines risk and threat assessment methods
  - Defines security testing strategy
  - Selects and provides security testing tools
  - Validates products
  - Defines and executes response process
- **Local security experts**
  - Embedded into development teams
  - Organize local security activities
  - Support developers and architects
  - Support product owners (responsibles)



# My Background

---

- I wear two hats:
  - Research Expert/Architect
  - **(Global) Security Testing Strategist**
- 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 SAP in a Nutshell
- 2 Motivation**
- 3 The Beginning: Large Scale Introduction of SAST
- 4 A Risk-based Security Testing Strategy
- 5 SAP's Secure Software Development Lifecycle (S<sup>2</sup>DL)
- 6 Myths and Lesson's Learned

# Costs of Vulnerabilities (Attacks on IT Systems)

---

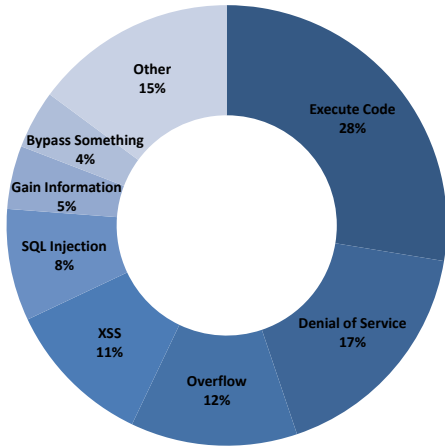
- TJX Company, Inc. (2007) \$ 250 million
- Sony (2011) \$ 170 million
- Heartland Payment Systems (2009) \$ 41 million

“

A hack not only costs a company money, but also its **reputation** and the **trust** of its customers. It can take years and millions of dollars to repair the damage that a single computer hack inflicts.

(<http://financialedge.investopedia.com/financial-edge/0711/Most-Costly-Computer-Hacks-Of-All-Time.aspx>)

# Vulnerability Types of CVE Reports Since 1999



- Causes for most vulnerabilities are
  - programming errors
  - configuration errors
- Patching
  - is expensive
  - may introduce new bugs
- How can we help developers to avoid this mistakes?

# Agenda

---

- 1 SAP in a Nutshell
- 2 Motivation
- 3 The Beginning: Large Scale Introduction of SAST**
- 4 A Risk-based Security Testing Strategy
- 5 SAP's Secure Software Development Lifecycle (S<sup>2</sup>DL)
- 6 Myths and Lesson's Learned

# How We Started: What We Wanted to Find

## Programming Patterns That May Cause Security Vulnerabilities

### Mainly two patterns

Local issues (no data-flow dependency), e.g.,

- Insecure functions

```
1 var x = Math.random();
```

- Secrets stored in the source code

```
1 var password = 'secret';
```

Data-flow related issues, e.g.,

- Cross-site Scripting (XSS)

```
1 var docref = document.location.href;
2 var input = docref.substring(
3     docref.indexOf("default=")+8);
4 var fake = function (x) {return x;}
5 var cleanse = function (x) {
6     return 'hello_world';}
7 document.write(fake(input));
8 document.write(cleanse(uinput));
```

- Secrets stored in the source code

```
1 var foo = 'secret';
2 var x = decrypt(foo,data);
```

# How We Started: What We Wanted to Find

## Programming Patterns That May Cause Security Vulnerabilities

### Mainly two patterns

Local issues (no data-flow dependency), e.g.,

- Insecure functions

```
1 var x = Math.random();
```

- Secrets stored in the source code

```
1 var password = 'secret';
```

Data-flow related issues, e.g.,

- Cross-site Scripting (XSS)

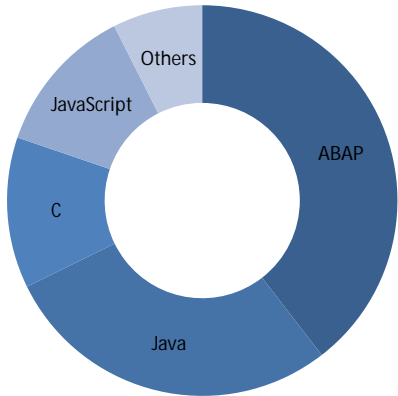
```
1 document.write(document.location.href);  
2 document.write(document.location.href.substring(  
3     document.location.href.indexOf("default=")+8));  
4  
5 fake = function (x) {return x;}  
6 var cleanse = function (x) {  
7     return 'hello_world';}  
8 document.write(fake(input));  
9 document.write(cleanse(uinput));
```

We trust our developers, i.e., we are focusing on finding "obvious" bugs.

- Secrets stored in the source code

```
1 var foo = 'secret';  
2 var x = decrypt(foo,data);
```

# SAST at SAP



- Since 2010, mandatory for all SAP products
- Multiple billions lines analyzed
- Constant improvement of tool configuration
- SAST tools used at SAP:

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

- Further details:  
Deploying Static Application Security Testing on a Large Scale. In GI Sicherheit 2014. Lecture Notes in Informatics, 228, pages 91-101, GI, 2014.

# So Everything is Secure Now, Right?

---

“

Our tool reports all vulnerabilities in your software – you only need to fix them and you are secure.

Undisclosed sales engineer from a SAST tool vendor.



# So Everything is Secure Now, Right?

---

“

Our tool reports all vulnerabilities in your software – you only need to fix them and you are secure.

Undisclosed sales engineer from a SAST tool vendor.

**Yes, this tools exists!** It is called Code Assurance Tool (cat):

# So Everything is Secure Now, Right?

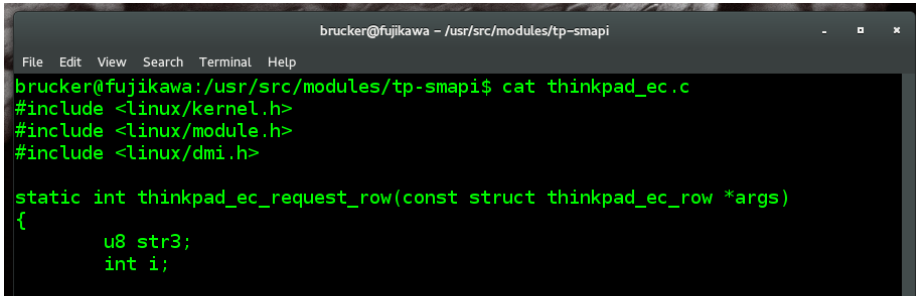
“

Our tool reports all vulnerabilities in your software – you only need to fix them and you are secure.

Undisclosed sales engineer from a SAST tool vendor.

**Yes, this tool exists!** It is called Code Assurance Tool (cat):

- The cat tool reports each line, that might contain a vulnerability:

A screenshot of a terminal window with a dark background and light green text. The window title is 'brucker@fujikawa - /usr/src/modules/tp-smapi'. The terminal shows the command 'cat thinkpad\_ec.c' and its output, which is the content of the file 'thinkpad\_ec.c'. The output includes include statements for 'linux/kernel.h', 'linux/module.h', and 'linux/dmi.h', followed by the start of a function definition 'thinkpad\_ec\_request\_row'.

```
brucker@fujikawa - /usr/src/modules/tp-smapi
File Edit View Search Terminal Help
brucker@fujikawa:/usr/src/modules/tp-smapi$ cat thinkpad_ec.c
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/dmi.h>

static int thinkpad_ec_request_row(const struct thinkpad_ec_row *args)
{
    u8 str3;
    int i;
```

# So Everything is Secure Now, Right?

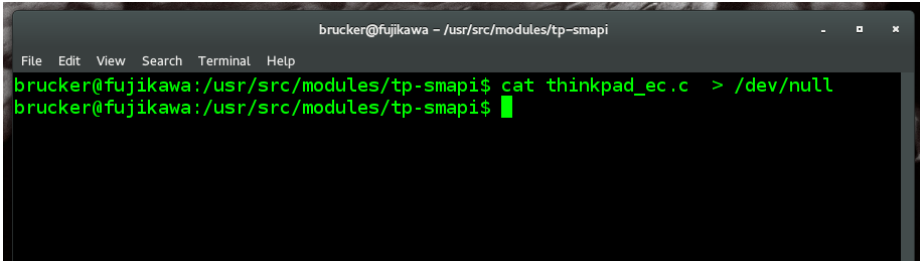
“

Our tool reports all vulnerabilities in your software – you only need to fix them and you are secure.

Undisclosed sales engineer from a SAST tool vendor.

**Yes, this tool exists!** It is called Code Assurance Tool (cat):

- The cat tool reports each line, that might contain a vulnerability:
- It supports also a mode that reports **no false positives**:

A terminal window titled 'brucker@fujikawa - /usr/src/modules/tp-smapi'. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal shows the following text:

```
brucker@fujikawa:/usr/src/modules/tp-smapi$ cat thinkpad_ec.c > /dev/null  
brucker@fujikawa:/usr/src/modules/tp-smapi$ █
```

# Agenda

---

- 1 SAP in a Nutshell
- 2 Motivation
- 3 The Beginning: Large Scale Introduction of SAST
- 4 A Risk-based Security Testing Strategy**
- 5 SAP's Secure Software Development Lifecycle (S<sup>2</sup>DL)
- 6 Myths and Lesson's Learned

# Combining Multiple Security Testing Methods and Tools

---



Client Application

Web Browser

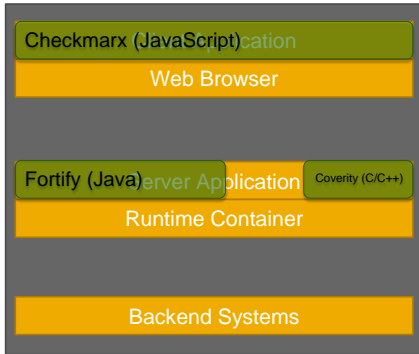
Server Application

Runtime Container

Backend Systems

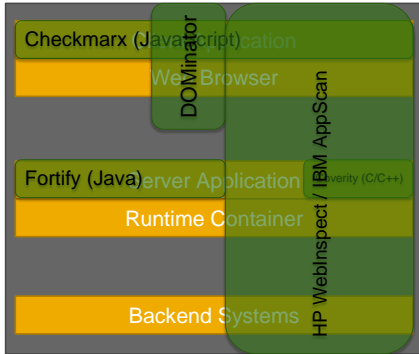
- Risks of only using only SAST
  - Wasting effort that could be used more wisely elsewhere
  - Shipping insecure software
- Examples of SAST limitations
  - Not all programming languages supported
  - Covers not all layers of the software stack

# Combining Multiple Security Testing Methods and Tools



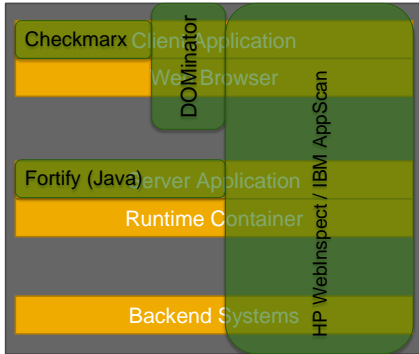
- Risks of only using only SAST
  - Wasting effort that could be used more wisely elsewhere
  - Shipping insecure software
- Examples of SAST limitations
  - Not all programming languages supported
  - Covers not all layers of the software stack

# 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



- 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

# Agenda

---

- 1 SAP in a Nutshell
- 2 Motivation
- 3 The Beginning: Large Scale Introduction of SAST
- 4 A Risk-based Security Testing Strategy
- 5 SAP's Secure Software Development Lifecycle (S<sup>2</sup>DL)**
- 6 Myths and Lesson's Learned

# SAP' Secure Software Development Lifecycle (S<sup>2</sup>DL)

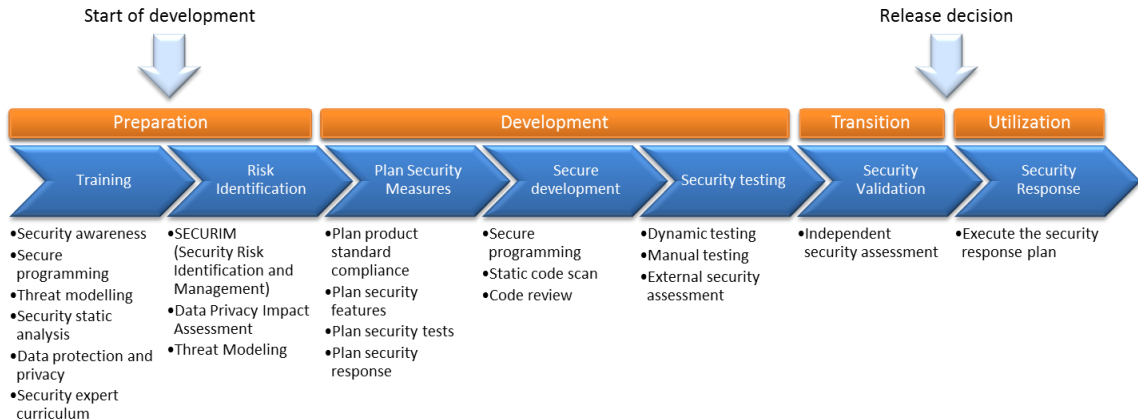
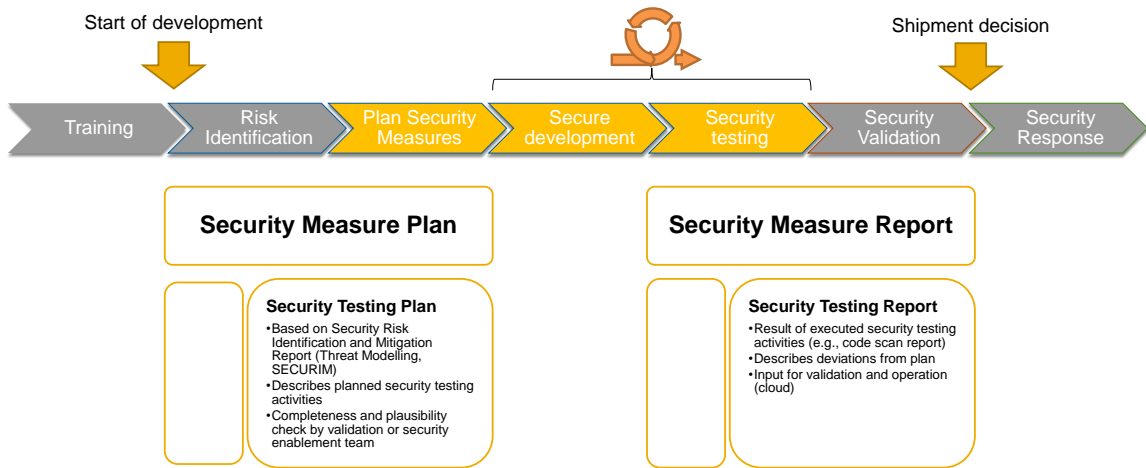


Figure: SAP SSDL

# SAP' Secure Software Development Lifecycle (S<sup>2</sup>DL)

## Security Testing Plan and Security Testing Report



# Agenda

---

- 1 SAP in a Nutshell
- 2 Motivation
- 3 The Beginning: Large Scale Introduction of SAST
- 4 A Risk-based Security Testing Strategy
- 5 SAP's Secure Software Development Lifecycle (S<sup>2</sup>DL)
- 6 Myths and Lesson's Learned**

# Continuously Measure Your Work and Improve Your Setup

## But How to Measure and What to Expect?

---

### What we do:

- Externally reported vulnerabilities/found by validation: check why we missed it earlier
- Potential reasons for missing a vulnerability (and actions)
  - Vulnerability not detected by our tools (strategy)
    - could be detected in principle by our tools  
⇒ analyze necessary changes (with tool vendor) and decide if risk justifies effort for enhancing tool
    - cannot be detected in principle by our tools  
⇒ research for suitable tools and and decide if risk justifies effort for introducing new tool
  - Vulnerability can be detected by our tools
    - With recent configuration but not configuration at release date  
⇒ no immediate actions necessary
    - With configuration at release date  
⇒ analyze why it was not detected and take further actions

### What we expect

- Issues not covered by current tool configuration should increase (ideally to 100%)

### What we observe

- Increase of logic-based flaws

# Penetration Tests at the End of Development

... test/ensure the security of the developed product, right?

---

Main purpose of penetration tests at end of development is:

- to check for “flaws” in the the S<sup>2</sup>DL (and not the product)
- Ideally, they only find:
  - no issues that can be fixed/detected earlier (e.g., configuration)

**Note**, penetration tests in productive environments are different:

- They test the actual configuration
- They test the productive environment (e.g., cloud/hosting)

# False Positives are not Your Biggest Concern

A Pragmatic Solution for Too Many Findings: Prioritize Them

Filter Set: SAP  My Issues

171 96 640 195 1102

**Corporate Security Requirements (171)**

Group By: Category

- ▶ Command Injection - [0 / 5]
- ▶ Cross-Site Scripting: Persistent - [0 / 38]
- ▶ Cross-Site Scripting: Reflected - [0 / 70]
- ▶ Dynamic Code Evaluation: Code Injection - [0 / 1]
- ▶ Header Manipulation - [0 / 7]
- ▶ Password Management: Empty Password - [0 / 2]
- ▶ Path Manipulation - [0 / 5]
- ▶ SQL Injection - [0 / 43]

- What needs to be audited
- What needs to be fixed
  - as security issue (response effort)
  - quality issue
- Different rules for
  - old code
  - new code



# False Positives are not Your Biggest Concern

A Pragmatic Solution for Too Many Findings: Prioritize Them

The screenshot shows a web-based interface for managing security audit findings. At the top, there is a 'Filter Set' dropdown menu set to 'SAP' and a checkbox for 'My Issues'. Below this is a row of colored buttons representing different severity levels: 171 (black), 96 (red), 640 (orange), 195 (grey), and 1102 (green). The red button is selected, and a red bar below it reads 'Audit All (default) (96)'. A 'Group By' dropdown menu is set to 'Category'. The main area displays a list of categories with expandable folders and counts:

- ▶ Insecure Randomness - [0 / 1]
- ▶ J2EE Bad Practices: Non-Serializable Object Stored in Se
- ▶ Null Dereference - [0 / 8]
- ▶ Password Management: Hardcoded Password - [0 / 3]
- ▶ Password Management: Password in Configuration File
- ▶ Privacy Violation - [0 / 45]
- ▶ Race Condition: Singleton Member Field - [0 / 1]
- ▶ Race Condition: Static Database Connection - [0 / 2]

- What needs to be audited
- What needs to be fixed
  - as security issue (response effort)
  - quality issue
- Different rules for
  - old code
  - new code

# False Positives are not Your Biggest Concern

A Pragmatic Solution for Too Many Findings: Prioritize Them

The screenshot shows a software interface for managing security findings. At the top, there is a 'Filter Set' dropdown menu set to 'SAP' and a checkbox for 'My Issues'. Below this is a summary bar with five colored boxes representing different severity levels: 171 (black), 96 (red), 640 (orange), 195 (grey), and 1102 (green). The orange box, representing 640 findings, is highlighted with a dashed border. Below the summary bar is a section titled 'Spot Checks of Each Category (640)' in an orange box. Underneath, there is a 'Group By:' dropdown menu set to 'Category'. The main area displays a list of categories, each with a folder icon and a count in brackets: 'Access Control: Database - [0 / 33]', 'Code Correctness: Erroneous Class Compare - [0 / 1]', 'Code Correctness: Erroneous String Compare - [0 / 4]', 'Cookie Security: Cookie not Sent Over SSL - [0 / 4]', 'Cross-Site Request Forgery - [0 / 27]', 'Denial of Service - [0 / 7]', 'Hidden Field - [0 / 15]', and 'J2EE Bad Practices: getConnection() - [0 / 5]'. A vertical scrollbar is visible on the right side of the list.

- What needs to be audited
- What needs to be fixed
  - as security issue (response effort)
  - quality issue
- Different rules for
  - old code
  - new code

# False Positives are not Your Biggest Concern

A Pragmatic Solution for Too Many Findings: Prioritize Them

Filter Set: SAP  My Issues

171 96 640 195 1102

Optional (195)

Group By: Category

- Axis 2 Misconfiguration: Debug Information - [0 / 6]
- Dead Code: Unused Method - [0 / 2]
- J2EE Bad Practices: Leftover Debug Code - [0 / 4]
- J2EE Bad Practices: Sockets - [0 / 1]
- J2EE Bad Practices: Threads - [0 / 6]
- J2EE Misconfiguration: Excessive Servlet Mappings - [0 / 6]
- J2EE Misconfiguration: Missing Data Transport Constraints - [0 / 6]
- Object Model Violation: Just one of equals() and hashCode()

- What needs to be audited
- What needs to be fixed
  - as security issue (response effort)
  - quality issue
- Different rules for
  - old code
  - new code

# Listen to Your Developers: Development Awareness

Developers Should be the Best Friends of Security Experts (not Their Enemies)

---

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

Always keep in mind:

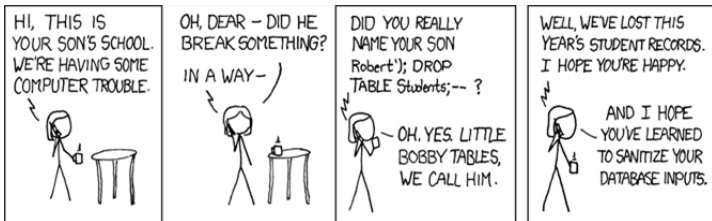
Building a a secure system more difficult than finding a successful attack.

We need:

- Easier to use security APIs
- More tools that make it easy to implement system securely
- Frameworks that make it hard to implement insecure systems
- ...

And, btw, this also holds for DevOps (Cloud)

# Thank you!



<http://xkcd.com/327/>

# Related Publications

---



**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, Lukas Brügger, and Burkhart Wolff.**

**Formal firewall conformance testing: An application of test and proof techniques.**

*Software Testing, Verification & Reliability (STVR)*, 25(1):34–71, 2015.

<http://www.brucker.ch/bibliography/abstract/brucker.ea-formal-fw-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. ISBN 978-3-88579-622-0.

<http://www.brucker.ch/bibliography/abstract/brucker.ea-sast-experiences-2014>.



**Achim D. Brucker and Burkhart Wolff.**

**On theorem prover-based testing.**

*Formal Aspects of Computing (FAC)*, 25(5):683–721, 2013.

ISSN 0934-5043.

<http://www.brucker.ch/bibliography/abstract/brucker.ea-theorem-prover-2012>.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE. The information contained herein may be changed without prior notice.

Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Excel, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, System i, System i5, System p, System p5, System x, System z, System z10, System z9, z10, z9, iSeries, pSeries, xSeries, zSeries, eServer, z/VM, z/OS, i5/OS, S/390, OS/390, OS/400, AS/400, S/390 Parallel Enterprise Server, PowerVM, Power Architecture, POWER6+, POWER6, POWER5+, POWER5, POWER, OpenPower, PowerPC, BatchPipes, BladeCenter, System Storage, GPFS, HACMP, RETAIN, DB2 Connect, RACF, Redbooks, OS/2, Parallel Sysplex, MVS/ESA, AIX, Intelligent Miner, WebSphere, Netfinity, Tivoli and Informix are trademarks or registered trademarks of IBM Corporation.

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

Adobe, the Adobe logo, Acrobat, PostScript, and Reader are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

SAP, R/3, SAP NetWeaver, Duet, PartnerEdge, ByDesign, SAP BusinessObjects Explorer, StreamWork, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE in Germany and other countries.

Business Objects and the Business Objects logo, BusinessObjects, Crystal Reports, Crystal Decisions, Web Intelligence, Xcelsius, and other Business Objects products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Business Objects Software Ltd. Business Objects is an SAP company.

Sybase and Adaptive Server, iAnywhere, Sybase 365, SQL Anywhere, and other Sybase products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Sybase, Inc. Sybase is an SAP company.

All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

The information in this document is proprietary to SAP. No part of this document may be reproduced, copied, or transmitted in any form or for any purpose without the express prior written permission of SAP SE.

This document is a preliminary version and not subject to your license agreement or any other agreement with SAP. This document contains only intended strategies, developments, and functionalities of the SAP® product and is not intended to be binding upon SAP to any particular course of business, product strategy, and/or development. Please note that this document is subject to change and may be changed by SAP at any time without notice.

SAP assumes no responsibility for errors or omissions in this document. SAP does not warrant the accuracy or completeness of the information, text, graphics, links, or other items contained within this material. This document is provided without a warranty of any kind, either express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.

SAP shall have no liability for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials. This limitation shall not apply in cases of intent or gross negligence.

The statutory liability for personal injury and defective products is not affected. SAP has no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third-party Web pages nor provide any warranty whatsoever relating to third-party Web pages.