SAP Code Vulnerability Analyzer

Peter Barker, Product Management
SAP SE, 2020
Disclaimer

The information in this presentation is confidential and proprietary to SAP and may not be disclosed without the permission of SAP. Except for your obligation to protect confidential information, this presentation is not subject to your license agreement or any other service or subscription agreement with SAP. SAP has no obligation to pursue any course of business outlined in this presentation or any related document, or to develop or release any functionality mentioned therein.

This presentation, or any related document and SAP's strategy and possible future developments, products and or platforms directions and functionality are all subject to change and may be changed by SAP at any time for any reason without notice. The information in this presentation is not a commitment, promise or legal obligation to deliver any material, code or functionality. This presentation 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. This presentation is for informational purposes and may not be incorporated into a contract. SAP assumes no responsibility for errors or omissions in this presentation, except if such damages were caused by SAP's intentional or gross negligence.

All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, which speak only as of their dates, and they should not be relied upon in making purchasing decisions.
The old-fashioned approach: expensive + reactive

1. Somebody builds insecure software
   - In-house
   - Outsourced
   - Commercial
   - Open source

2. IT deploys the insecure software

3. Breach or pen test proves our code is bad

4. We convince and pay developers to fix it

© 2020 SAP SE or an SAP affiliate company. All rights reserved.
You know the challenge – breaches are increasing

World’s largest data breaches and hacks

http://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/
Application security testing solutions at SAP

Dynamic application security testing
- Find vulnerabilities in the running application
- Manual application penetration testing
- Automated application vulnerability scanning

Static application security testing
- Find vulnerabilities analyzing the sources
- Manual source code review
- Automated source code analysis

SAP Code Vulnerability Analyzer (CVA)

Finding security issues at design time instead of in production is easier and less expensive!
Enterprise application security best practice from SAP

- SAP Development runs security tests on all SAP applications and code delivered by SAP.

- SAP Development uses SAP CVA to scan more than 500 million lines of ABAP code before delivery to our customers.
SAP Code Vulnerability Analyzer
as a part of the ABAP Test Cockpit (ATC)

SAP GUI
Eclipse (ADT)

Security (CVA)
Code robustness
SAP S/4HANA readiness
Custom checks
Unit tests
Performance
Usability

SAP GUI
Eclipse (ADT)

ABAP Test Cockpit (ATC)

Solution Manager

CCLM
CHARM

Extraction

SAP Code Vulnerability Analyzer
as a part of the ABAP Test Cockpit (ATC)

Checked System A (>=7.00)
Customer Code

Checked System B (>=7.00)
Customer Code

Checked System C (>=7.00)
Customer Code
Demo
Introductory example: SQL injection

```
REPORT zsqlin_demo_sql_injection_2.
PARAMETERS: street TYPE zemployees-street LOWER CASE, 
zipcode TYPE zemployees-zipcode LOWER CASE, 
city TYPE zemployees-city LOWER CASE, 
phone TYPE zemployees-phone_ext.
DATA: set_expr TYPE string,
     user TYPE xsbnames.

□ IF street IS NOT INITIAL. 
  set_expr = set_expr ++ ' STREET = ' ++ street ++ ' '.
END IF.
□ IF zipcode IS NOT INITIAL. 
  set_expr = set_expr ++ ' ZIPCODE = ' ++ zipcode.
END IF.
□ IF city IS NOT INITIAL. 
  set_expr = set_expr ++ ' CITY = ' ++ city ++ ' '.
END IF.
□ IF phone IS NOT INITIAL. 
  set_expr = set_expr ++ ' PHONE = ' ++ phone.
END IF.
□ IF set_expr IS NOT INITIAL. 
  user = cl_abap_systx-<user_name()>.
UPDATE zemployees 
  SET (set_expr) 
  WHERE userid = user.
END IF.
```

Input for street: `xyz' salary = '1500`

set_expr:
STREET = 'xyz'
salary = '1500'

Possible SQL injection (SET clause)

... 
SET STREET = 'xyz'
salary = '1500'
How the code analysis works

The Code Analyzer is searching for potentially vulnerable statements, where the input comes from untrusted sources. Only such occurrences are reported!
Baseline in ATC – Focus on findings in new or recently changed code

Baseline concept
Transfer individual ATC results into the baseline

Options
• suppress all findings (exclude from the ATC result)
• indicate the findings as exempted
• assign the low priority to the findings

The baseline is effective as long as the related code sections remain unchanged

Baseline activities
Adding/removing individual check results
Deleting baseline (reset the test system to original state)

Baseline management

More about Working with Baseline in ATC
Priority of each check can be adjusted to match the requirements

- Ability to control the priority of every single finding
- Take into account your own risk and security requirements.
- Possibility of a phased approach, enabling security checks over time to have a higher acceptance by developers.
Use exemption workflow to deal with false positives
Remote code analysis with ATC

One central ATC system (SAP_BASIS >=7.51) for all security checks in your system landscape

Remote stubs return a model from custom code

Check logic is executed on central system

Check variant is maintained in central system

New checks are installed on central system

Exemptions are stored on central system

Integrated in the development system

→ Minimized administration efforts

→ One quality standard for your whole system landscape
SAP’s application security offering focuses on the product CVA:

<table>
<thead>
<tr>
<th>Material</th>
<th>Item</th>
<th>Coding</th>
<th>Blocks of</th>
<th>Pricing metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>7019502</td>
<td>SAP CVA</td>
<td>ABAP</td>
<td>5</td>
<td>Users</td>
</tr>
</tbody>
</table>
Security checks in detail

Overview of available checks
Overview of available checks

Security Checks

- SQL Injection (ABAP SQL)
- SQL Injection (ADBC)
- Code Injection (ABAP)
- Call Injection
- OS Command Injection
- Directory Traversal
- Backdoors & Authorizations
- Web Exploitation
Overview of the available checks
- SQL injection (ABAP SQL) -

**Manipulation of dynamic ABAP SQL**

- Potential manipulation of the dynamic WHERE condition (1101)
- Potential manipulation of a dynamic WHERE condition using the parameter I_FILTER of the object services method CREATE_QUERY (1122)
- Potential manipulation of the SET clause in the statement UPDATE (1112)
- Potential read performed on an illegal database table in a SELECT statement (1118)
- Potential read performed on an illegal database table in a modifying OpenSQL statement (1120)
- Potential read performed using an invalid secondary database connection in an Open SQL statement (1121)
- Potential read performed on invalid table columns (1114)
- Potential use of illegal columns in a dynamic GROUP BY clause (1116)
- Potential use of illegal columns in a dynamic HAVING clause (1117)
- Read performed on sensitive database table (11G0)
- Write performed on sensitive database table (11G1)
Overview of the available checks
- Backdoors & authorizations -

**Weak authorization checks or user administration bypassed**

- Hard-coded user name, possibly from undeleted test code or an indication of a back door (0821)
- Hard-coded host name sy-host, possibly from undeleted test code or an indication of a back door (11S1)
- Hard-coded system ID sy-sysid, possibly from undeleted test code or an indication of a back door (11S2)
- Hard-coded client sy-mandt, possibly from undeleted test code or an indication of a back door (11S3)
- System variable sy-xxxx compared with a hard-coded value from forgotten test code or that could indicate a back door (11S4).
- SY-SUBRC not evaluated after the statement AUTHORITY-CHECK (1160)
- SY-SUBRC not evaluated after switchable authorization check (1161)
- AUTHORITY-CHECK with explicit user name (1180)
- AUTHORITY-CHECK with explicitly specified user name sy-uname (1181)
- SY-SUBRC not handled after a security-relevant function was called (1165)
- Static CALL TRANSACTION without or with possibly insufficient authorization check (114A, 114B, 114C, 114D)
- FILTER addition of the statement OPEN DATASET used (1107)
- Potentially missing authorization check in a report (11A1)
- Potentially missing authorization check in an RFC function module (11A2)
Availability of SAP Code Vulnerability Analyzer – checking system

- SAP Code Vulnerability Analyzer is available as of:
  - SAP NetWeaver AS ABAP 7.4 Support Package 15 and later releases
  - SAP NetWeaver AS ABAP 7.5 including the new remote check framework with 7.51
CVA: The competition

The ABAP language developers are at SAP.

At SAP we use CVA to check our code.

We have over 200 CVA customers.

Integration: CVA is already in NetWeaver, it just needs to be activated – or is your solution running elsewhere, say, on a Java engine with all the overhead that it involves?

How does your solution handle updates? CVA’s central scanning approach minimizes the administrative work required to provide the latest CVA checks.

The number of checks is irrelevant – you need the right checks -> fewer false positives

Small company risk: Will the company still be around in 5 years’ time?
Proof of Concept of SAP CVA
Scan Results: Security Analyses in Extended Program Check

<table>
<thead>
<tr>
<th>Statistics: Check</th>
<th>Description</th>
<th>Prio 1</th>
<th>Prio 2</th>
<th>Prio 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Total</strong></td>
<td>Total</td>
<td>1,229</td>
<td>1,104</td>
<td>17,758</td>
</tr>
<tr>
<td><img src="Check" alt=" " /></td>
<td>Test Existence of a Program</td>
<td>Check Title</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td><img src="Check" alt=" " /></td>
<td>Security Analyses in Extended Program Check (SLIN)</td>
<td>Check Title</td>
<td>1,166</td>
<td>1,104</td>
</tr>
<tr>
<td><img src="Check" alt=" " /></td>
<td>Hard-coded user name</td>
<td>Check Message</td>
<td>72</td>
<td></td>
</tr>
<tr>
<td><img src="Check" alt=" " /></td>
<td>Potential SQL injection (WHERE condition)</td>
<td>Check Message</td>
<td>41</td>
<td></td>
</tr>
<tr>
<td><img src="Check" alt=" " /></td>
<td>Potential directory traversal</td>
<td>Check Message</td>
<td>846</td>
<td></td>
</tr>
<tr>
<td><img src="Check" alt=" " /></td>
<td>Potential ABAP command injection</td>
<td>Check Message</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td><img src="Check" alt=" " /></td>
<td>Potential SQL injection (column names)</td>
<td>Check Message</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td><img src="Check" alt=" " /></td>
<td>Potential SQL injection (table name when read performed)</td>
<td>Check Message</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td><img src="Check" alt=" " /></td>
<td>Potential SQL injection (table name when write performed)</td>
<td>Check Message</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td><img src="Check" alt=" " /></td>
<td>User-driven dynamic RFC function module call</td>
<td>Check Message</td>
<td>167</td>
<td></td>
</tr>
<tr>
<td><img src="Check" alt=" " /></td>
<td>User-driven dynamic procedure call of program unit</td>
<td>Check Message</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td><img src="Check" alt=" " /></td>
<td>User-driven dynamic LEAVE TO TRANSACTION</td>
<td>Check Message</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td><img src="Check" alt=" " /></td>
<td>Static CALL TRANSACTION without authorization check</td>
<td>Check Message</td>
<td>173</td>
<td></td>
</tr>
<tr>
<td><img src="Check" alt=" " /></td>
<td>Static CALL TRANSACTION without authorization check (restricted functions)</td>
<td>Check Message</td>
<td>1,057</td>
<td></td>
</tr>
<tr>
<td><img src="Check" alt=" " /></td>
<td>Dynamic CALL TRANSACTION without authorization check</td>
<td>Check Message</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td><img src="Check" alt=" " /></td>
<td>Dynamic CALL TRANSACTION with data flow and without authorization check</td>
<td>Check Message</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td><img src="Check" alt=" " /></td>
<td>AUTHORITY-CHECK without sy-subr handling</td>
<td>Check Message</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td><img src="Check" alt=" " /></td>
<td>System function call CALL 'SYSTEM'</td>
<td>Check Message</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td><img src="Check" alt=" " /></td>
<td>Read on sensitive database tables</td>
<td>Check Message</td>
<td>870</td>
<td></td>
</tr>
<tr>
<td><img src="Check" alt=" " /></td>
<td>Write on sensitive database tables</td>
<td>Check Message</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td><img src="Check" alt=" " /></td>
<td>Hard-coded host name</td>
<td>Check Message</td>
<td>59</td>
<td></td>
</tr>
<tr>
<td><img src="Check" alt=" " /></td>
<td>Hard-coded system ID</td>
<td>Check Message</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td><img src="Check" alt=" " /></td>
<td>Hard-coded client</td>
<td>Check Message</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td><img src="Check" alt=" " /></td>
<td>Prerequisites for the extended program check (SLIN)</td>
<td>Check Title</td>
<td>46</td>
<td></td>
</tr>
</tbody>
</table>
Further information

SAP Code Vulnerability analyzer

Documentation

SAP Community

Blogs
- One central check system for multiple systems on various releases
Thank you

Peter Barker
SAP Product Management, SAP SE
peter.barker@sap.com
No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. Please see http://global12.sap.com/corporate-en/legal/copyright/index.epx for additional trademark information and notices.

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

National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP SE or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP SE or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE’s or its affiliated companies’ strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, which speak only as of their dates, and they should not be relied upon in making purchasing decisions.