SAP NetWeaver Application Server Add-On for Code Vulnerability Analysis
Disclaimer

This presentation outlines our general product direction and should not be relied on in making a purchase decision. This presentation is not subject to your license agreement or any other agreement with SAP. SAP has no obligation to pursue any course of business outlined in this presentation or to develop or release any functionality mentioned in this presentation. This presentation and SAP's strategy and possible future developments are subject to change and may be changed by SAP at any time for any reason without notice. 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 assumes no responsibility for errors or omissions in this document, except if such damages were caused by SAP intentionally or grossly negligent.
Source Code
The Source of Security Risks
Business Applications Do Have a History

**Today's business applications**
- Have often evolved over several years
- Are complex
- Are built on changing requirements
- Were created based on different development paradigms
- Have been optimized for performance
- Have been extended, but not reinvented

And often, security was only an afterthought.
The Security Challenge

In order to **secure** an application, you have to understand **all** of its components, functions, infrastructure, and the related **threats**.

In order to **break** an application, just **one flaw** in any of its components/functions or the infrastructure may be enough.

**The problem:**
Each new technology brings about new vulnerabilities.
Firewalls, intrusion detection systems, signatures, and encryption alone cannot make an application secure.
Security testing, consisting of dynamic application security testing (DAST) and/or static application security testing (SAST) can improve code quality and security. Neither DAST nor SAST are guaranteed to find all security issues in an application.
Automated Detection of Weaknesses in ABAP Sources
SAP NetWeaver Code Vulnerability Analyzer Feature Set

- Integrated into standard ABAP development infrastructure for easy consumption by developers
- Reduced false-positive rate through data flow analysis
- Supports exemption workflows to ease handling of false positives
- Supports automation requirements by quality assurance teams
- Priority of each check can be adjusted to match the requirements
- Extensive documentation to support developers in fixing the detected issues

Increased security for your applications
Introductory Example: SQL Injection

```
REPORT zslin_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 xubname.

IF street IS NOT INITIAL.
  set_expr = set_expr && 'STREET = ' && street && '';.
ENDIF.

IF zipcode IS NOT INITIAL.
  set_expr = set_expr && 'ZIPCODE = ' && zipcode && '';.
ENDIF.

IF city IS NOT INITIAL.
  set_expr = set_expr && 'CITY = ' && city && '';.
ENDIF.

IF phone IS NOT INITIAL.
  set_expr = set_expr && 'PHONE = ' && phone && '';.
ENDIF.

IF set_expr IS NOT INITIAL.
  user = cl_abap_system->user_name( ).
  UPDATE zemployees
  SET (set_expr)
  WHERE userid = user.
ENDIF.
```

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

Possible SQL injection (SET clause)

```
... SET STREET = 'xyz'
    salary = '1500'
```
How Does Code Vulnerability Analysis Work?

The Code Analyzer searches for and reports potentially vulnerable statements where the input comes from untrusted sources.
Integrated Into Standard Developer Tools

Code checks are integrated into the ABAP Test Cockpit and can be launched easily from most developer tools such as SE80 or SE38.

You can launch checks for single objects as well as groups of objects.
Supporting Developers in Fixing Code

Detailed documentation of detected issues includes explanations as well as advice on avoiding vulnerabilities, helping developers understand and fix issues in their code.

The tool supports direct navigation to:

- the location in code sources
- the related documentation
- the workflow to create an exemption
Corrected Program

```
REPORT zslin_demo_sql_injection_2_fix.

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 xubname.

IF street IS NOT INITIAL.
  set_expr = set_expr " STREET = ' & cl_abap_dyn_prg=>quote( street ).
ENDIF.

IF zipcode IS NOT INITIAL.
  set_expr = set_expr " ZIPCODE = ' & cl_abap_dyn_prg=>quote( zipcode ).
ENDIF.

IF city IS NOT INITIAL.
  set_expr = set_expr " CITY = ' & cl_abap_dyn_prg=>quote( city ).
ENDIF.

IF phone IS NOT INITIAL.
  set_expr = set_expr " PHONE_EXT = ' & phone.
ENDIF.

IF set_expr IS NOT INITIAL.
  user = cl_abap_syst=>get_user_name( ).
  UPDATE zemployees
     SET (set_expr)
     WHERE userid = user.
ENDIF.
```
Identifying Common Source Code Risks of the OWASP Top 10

**Injection**
Injection flaws, such as SQL and OS injection occur when un-trusted data is sent to an interpreter as part of a command or query.

**Insecure Direct Object References**
A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.

**Missing Function Level Access Control**
Most applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed.

**Using Components with Known Vulnerabilities**
Components, such as libraries, frameworks, and other software modules, almost always run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover.
You can control the priority of every single finding, enabling you to consider your own risk and security requirements.

This allows for a phased approach: Security checks can be extended over time to increase developer acceptance.
Integration into the ABAP Test Cockpit (ATC)

ATC is an ABAP check framework which allows developers to run static checks and unit tests for ABAP programs. ATC is fully integrated into the development environment and transport tools, and offers features such as instant navigation, documentation, and fix recommendation.

What are the benefits?

- ATC is the single point of entry for all static code check tools
- ATC includes a 4-eye principle exemption process to handle findings effectively
- ATC is fully integrated in the ABAP development workbench and offers high usability for developers and quality experts
- ATC is not only a check tool; in addition, it supports essential QA techniques such as Q gates or regression testing in a consolidation system
Testing ABAP Everywhere with the ABAP Test Cockpit (ATC)
ABAP Test Cockpit (ATC)

What is it?

- ATC is an ABAP check framework for running static checks and unit tests for ABAP programs
- ATC is fully integrated into the development environment and transport tools, along with instant navigation, documentation, and fix recommendation

What are the benefits?

- ATC is the single point of entry for all static code check tools
- ATC includes a 4-eye principle exemption process to handle findings effectively
- ATC is fully integrated in the ABAP development workbench and offers high usability for developers and quality experts
- ATC is not only a check tool; in addition, it supports essential QA techniques such as Q gates or regression testing in a consolidation system
ABAP Test Cockpit Integrated into the ABAP IDE
Example Development Landscape

- Development System 1
  - Developers run static/unit/scenario tests on their objects
  - Periodic checks run to validate the code of a development team
  - Quality-Gate: Check during transport release

- Development System 2

- Consolidation System
  - Q-experts run mass checks and distribute the results
  - Quality-Gate: Mass check run and consolidation test

- Use ONE quality standard for Q-Gates
Security Checks for Development on Older Releases

When development systems are older than the minimal supported release, you can use a security test system to execute tests on a higher release.
Features for Developers

ATC Features

☑ Start ATC within different ABAP workbench tools: SE80, SE24, SE38, SE11…
☑ ATC runs automatically during release of transport requests
☑ Easy access to central ATC results in the development systems
☑ User-centric display of ATC results, including powerful filter, navigation, and re-check…

Checks code during development and transport release
Corrects bugs
Requests exemptions for false-positives
Features for Quality Experts

ATC Features

- Exemption approval process
- E-mail ATC result to responsible contact person
- Statistics show aggregation of ATC findings using different criteria
- Execution of ABAP unit tests

Defines commonly used check variant
Monitors quality of the whole code base
Approves exemptions
ATC Administrator

ATC Features

- Powerful parallelization engine to run mass tests very effectively
- Restart capability in case of a canceled/crashed ATC run
- Possibility to schedule regular ATC runs
- Powerful monitoring tool and flexible logging
- Distribute ATC results to multiple target systems (e.g. from consolidation to development systems)
Security Checks in Detail
Overview of Available Checks

- SQL Injection (Open SQL)
- SQL Injection (ADBC)
- Code Injection (ABAP)
- Call Injection
- OS Command Injection
- Directory Traversal
- Backdoors & Authorizations
- Web Exploitability
Overview of Available Checks
SQL Injection (Open SQL)

Manipulation of Dynamic Open 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 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)
Overview of Available Checks
SQL Injection (ADBC)

Manipulation of SQL Statements

- Potential injection of harmful SQL statements of clauses in execution of DDL statements in ADBC (1128)
- Potential injection of harmful SQL statements of clauses in execution of DML statements in ADBC (1130)
Overview of Available Checks
Code Injection (ABAP)

Manipulation of Dynamically Created ABAP Code

- Potential injection of harmful code in the statements INSERT REPORT and GENERATE SUBROUTINE POOL (1108)
- Potential manipulation of the dynamic WHERE condition in an internal table (1190)
Overview of Available Checks
Call Injection

Manipulation in Dynamic Calls
- Potential call of an illegal transaction using the statement CALL TRANSACTION (1142)
- Potential call of an unwanted transaction using the statement LEAVE TO TRANSACTION (1143)
- Potential call of an illegal program using the statement SUBMIT (1141)
- Potential call of invalid function module using RFC (1140)
Overview of Available Checks
OS Command Injection

Injections of Operating System Commands

- Statement CALL 'SYSTEM' used (1170)
- Potential manipulation in the FILTER addition of the statement OPEN DATASET (1106)
Overview of Available Checks
Directory Traversal

Access to Illegal Directories and Files

- Potential manipulation of the file name in the statement OPEN DATASET or DELETE DATASET (1104)
- Potential manipulation of the file name in the method CREATE_UTF8_FILE_WITH_BOM of the class CL_ABAP_FILE_UTILITIES (1124)
Overview of 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)
- SY-SUBRC not evaluated after the statement AUTHORITY-CHECK (1160)
- AUTHORITY-CHECK with explicit user name (1180)
- AUTHORITY-CHECK with explicitly specified user name sy-uname (1181)
Overview of Available Checks
Web Exploitability

Possible Attacks Using Web Technologies
- Obsolete escape method used (1150)
Summary
Writing Secure ABAP Code

One weakness is enough to put your business at a risk!

- Regularly check your source code and ensure that your code is compliant to state-of-the-art security programming best practices.
- Train developers to raise awareness about common weaknesses.
- Security is not a one-off project. Incorporate security improvements into your development routines.
Summary: Code Vulnerability Analyzer

- Developed by the team that creates the ABAP language
- Tightly integrated into standard testing infrastructure
- Thoroughly tested and used by SAP internally for several years
- Successfully piloted by customers
- SAP NetWeaver Application Server, add-on for code vulnerability analysis is planned to be available as of:
  - SAP NetWeaver AS ABAP 7.0 EhP2 Support Package 14
  - SAP NetWeaver AS ABAP 7.0 EhP3 Support Package 09
  - SAP NetWeaver AS ABAP 7.3 EhP1 Support Package 09
  - SAP NetWeaver AS ABAP 7.4 Support Package 05 and later releases
Summary: ABAP Test Cockpit

✓ ATC is the standard ABAP check framework at SAP
✓ The ABAP Test Cockpit (ATC) is a tool for doing static and dynamic quality checks of ABAP code and associated repository objects
✓ ATC is based on Code Inspector → Very easy migration: Just re-use your current global Code Inspector check variant
✓ ATC is available as part of:
  - SAP NetWeaver AS ABAP 7.0 EhP2 Support Package 12
  - SAP NetWeaver AS ABAP 7.0 EhP3 Support Package 05
  - SAP NetWeaver AS ABAP 7.3 EhP1 Support Package 05
  - SAP NetWeaver AS ABAP 7.3 EhP2 and later releases
Further Information

SAP NetWeaver Application Server, add-on for code vulnerability analysis
  • http://wiki.scn.sap.com/wiki/display/ABAP/SAP+NetWeaver+Application+Server%2C+add-on+for+code+vulnerability+analysis

ABAP Test and Analysis Tools
  • http://wiki.sdn.sap.com/wiki/display/ABAP/ABAP+Test+and+Analysis+Tools

SAP Code Inspector

ABAP Test Cockpit (ATC)
  • http://wiki.sdn.sap.com/wiki/display/ABAP/ABAP+Test+Cockpit

SAP Community
  • http://scn.sap.com/community/security
  • http://scn.sap.com/community/abap/testing-and-troubleshooting
No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

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

National product specifications may vary.

These materials are provided by SAP AG and its affiliated companies (“SAP Group”) for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group 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.

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