Today’s business applications are often complex programs built over a number of years to help companies run as efficiently as possible. Organizations often pay a great deal of attention to fine-tuning these applications for performance, but even the highest levels of performance cannot protect the sensitive assets hosted by these applications. Securing data is not only in the interest of that data’s business owners — for example, it is critical to HR’s purposes to secure HR data — but is also necessary to meet legal requirements and industry standards.

Cryptography, authentication, and authorizations are fundamental parts of protecting your data assets, but even more important is ensuring that the code that is driving your applications executes correctly and efficiently without opening up opportunities for attackers. SAP ensures the security of its own application code by following a standard set of test procedures that include an integrated set of static and dynamic security checks and manual reviews. (For more on the different types of checks and reviews that help ensure application security, see the online version of this article at SAPinsiderOnline.com.) To help its customers follow the same approach to ensure the security of their own applications, SAP provides a comprehensive, integrated framework of testing tools and checks as a part of SAP NetWeaver AS ABAP.

While the standard tools and checks that SAP provides for customers cover a wide range of the testing needed to ensure application security, an integrated tool for efficiently scanning ABAP source code for security vulnerabilities has not been available — until now. SAP now offers to customers the tool it uses internally to scan its own ABAP source code for security risks: SAP NetWeaver AS, add-on for code vulnerability analysis.

### Eliminating Exposure at the Source

SAP NetWeaver AS, add-on for code vulnerability analysis scans ABAP source code for...

---

the critical flaws identified by the Open Web Application Security Project (OWASP) for 2013 (see Figure 1). In particular, it focuses on:

- Injection attacks (A1), such as SQL, OS, or code injections
- Insecure direct object references (A4), such as directory traversal attacks
- Missing access control (A7)
- Insecure use of SAP NetWeaver AS ABAP functions (A9)

The security checks of SAP NetWeaver AS, add-on for code vulnerability analysis are more precise than the standard checks included with SAP NetWeaver AS ABAP because they are able to use information about the application’s internal data flow — such as how data entered into a field in a user interface is used within an ABAP statement — to identify risks. In addition, the security checks do not show warnings or errors when they detect coded security precautions, such as sanitizations or secure sources of data, thereby increasing efficiency by avoiding time wasted on false alarms.

Instead, the search focuses on places in the code where input from outside of the code unit is used in a statement, which is known to be a potentially insecure location within the code.

How It Works
SAP NetWeaver AS, add-on for code vulnerability analysis is tightly integrated into the existing test infrastructure of SAP NetWeaver AS ABAP. This infrastructure is based on the Code Inspector, which is a test framework that comprises a variety of development checks and functions for evaluating code elements such as syntax, performance, security, and reliability. Although SAP NetWeaver AS, add-on for code vulnerability analysis can be used standalone, SAP recommends using it via the ABAP Test Cockpit. (See the sidebar for more on the ABAP Test Cockpit.)

You enable the extended security checks of SAP NetWeaver AS, add-on for code vulnerability analysis in your SAP NetWeaver AS ABAP system using the program RSLIN_SEC_LICENSE_SETUP. Once enabled, it is recommended that you adapt the default check variant for the ABAP Test Cockpit to contain the extended security checks, so that they are automatically included when you run the analysis. This is done in the Code Inspector (transaction SCI) by enabling the option “Security Analyses in Extended Program Check” (see Figure 2).

Now a developer can launch the extended security checks directly from within the developer environment, including the ABAP Workbench as well as the ABAP development tools for Eclipse. In the ABAP Editor (transaction SE38), for instance, the developer can launch the checks for a program, function, class, or package by simply right-clicking on the desired object and selecting Check > ABAP Test Cockpit (see Figure 3). The ABAP Test Cockpit will then run the checks and present the results in a flexible list that allows for grouping or filtering of entries.

Example Scenario: Identifying and Securing a Vulnerability
Let’s see what it looks like when the ABAP Test Cockpit tool identifies a risk after running the extended security checks of SAP NetWeaver AS, add-on for code vulnerability analysis, and what kind of support is available to help developers address any identified issues.
Identifying a Vulnerability

Figure 4 shows the source code for a simple self-service application that allows users to maintain addresses (street name, ZIP code, and city name) within the SAP NetWeaver AS ABAP system.

The extended security checks will not only look for a specific statement — such as an ABAP “update” statement — to raise a security message, but will also look for statements used in an insecure fashion (for example, statements that allow unsecured user input). In the example in Figure 4, the ABAP statement UPDATE (line 21) could be potentially misused, as it uses a dynamic SET condition. The dynamic part of an ABAP statement is usually written with an identifier (a variable name) enclosed by parentheses; in the example, (set_expr) in line 22. This code will lead to set_expr being evaluated by the ABAP interpreter during the runtime of the program and then executed by the ABAP application server. Assuming there is a field with the name salary, a correct set expression would look like:

```
STREET = 'xyz' salary = '1500'
```

In general, a dynamic statement is not always a risk. It becomes a risk if it allows the user of the program to exploit the data within the set_expr variable. To identify such risks, the extended security checks search for user input (in the example, the PARAMETERS statement in line 3) that is used in a potentially dangerous statement (the UPDATE statement in line 21) either directly or indirectly (in the example, it is used indirectly as part of the dynamic statement). The extended security checks recognize the data flow of the parameter street (declared in line 3) and the assignment to the set_expr (line 10), and will raise a security finding for this code (in this case, the vulnerability is a possible SQL injection). The data flow is presented in the ABAP Test Cockpit message (see Figure 5 on the next page), allowing the developer to understand the data flow from the source to the dangerous statement.

User input could reach a potentially dangerous statement from a variety of channels. A PARAMETER statement is only one example of such sources; others could be the interface of a remote function module or database tables in which the user can create or change data. Depending on the data type, not all of these input channels are
considered exploitable for possible attackers. To avoid false warning messages, the extended security checks also consider the structure of the data. For example, it’s very difficult for an attacker to exploit a set_expr that allows the user to enter only integers, so in the case of the example, the checks would not identify the set_expr for the ZIP code (line 13) as a vulnerability.

Securing a Vulnerability
Once the extended security checks identify a risk, how does a developer go about fixing it? Detailed online documentation is provided for the ABAP developer for each security check performed by the ABAP Test Cockpit. To access the documentation, the developer clicks on the message displayed in the ABAP Test Cockpit (refer back to Figure 5).

The documentation explains the risk that has been identified for the example scenario (see Figure 6), and provides advice on how the code should be changed to ensure proper, secure use of the program. It also describes the used procedure of the security checks, so that the developer can understand why the checks are reporting an issue.

Using the information from the online documentation, the developer can adapt the code, as shown in Figure 7. For the example, adding a quote function in lines 10 and 16 securely encodes the parameters. Because the ZIP code in this example is an integer field, which means that an attacker cannot exploit this field in the way a free-text field can be exploited, the encoding function is not necessary for the zipcode parameter.

With these changes in place, the security checks will no longer raise an error message for this code.

Summary
Your valuable data assets are only as secure as the applications that host and access them. SAP is committed to providing its customers with the same high-quality checks and testing capabilities it trusts to secure its own software, as well as the guidance customers need to put in place a secure approach to coding applications for SAP NetWeaver AS ABAP. With the release of SAP NetWeaver AS, add-on for code vulnerability analysis, SAP brings to its customers the powerful scanning functionality it has used for years to identify vulnerabilities in the source code of its own applications, and it has already been piloted in a recent customer engagement initiative with much success. Backed by the knowledge of the team that develops the ABAP language, SAP NetWeaver AS, add-on for code vulnerability analysis enables you to easily and efficiently locate security risks in your code, so you can create secure applications with confidence.


3 The secure programming guide for ABAP provides documentation on how customers can achieve a high level of security for their applications on SAP NetWeaver AS ABAP. The guide is available at http://bit.ly/15K5zn.