Protecting SAP® Applications Based on Java and ABAP™ Against Common Attacks
Version 1.2
November 2011
# Table of Contents

<table>
<thead>
<tr>
<th>Page</th>
<th>Section</th>
<th>Vulnerabilities</th>
<th>Countermeasures</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Introduction</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>How to Read This Document</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Background</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Summary</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Transport Layer Security Using HTTPS</td>
<td>Vulnerabilities</td>
<td>Countermeasures</td>
</tr>
<tr>
<td>8</td>
<td>Secure Session Handling</td>
<td>Vulnerabilities</td>
<td>Countermeasures</td>
</tr>
<tr>
<td>10</td>
<td>Cross-Site Request Forgery (XSRF)</td>
<td>Vulnerabilities</td>
<td>Countermeasures</td>
</tr>
<tr>
<td>12</td>
<td>SQL Injection</td>
<td>Vulnerabilities</td>
<td>Countermeasures</td>
</tr>
<tr>
<td>13</td>
<td>Directory Traversal</td>
<td>Vulnerabilities</td>
<td>Countermeasures</td>
</tr>
<tr>
<td>14</td>
<td>Cross-Site Scripting</td>
<td>Vulnerabilities</td>
<td>Countermeasures</td>
</tr>
<tr>
<td>16</td>
<td>Invoker Servlet</td>
<td>Vulnerability</td>
<td>Countermeasures</td>
</tr>
<tr>
<td>17</td>
<td>ABAP Code Injection</td>
<td>Vulnerabilities</td>
<td>Countermeasures</td>
</tr>
<tr>
<td>18</td>
<td>Hard-Coded User Names</td>
<td>Vulnerabilities</td>
<td>Countermeasures</td>
</tr>
<tr>
<td>19</td>
<td>Appendix 1 – Overview Table</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Appendix 2 – References</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Introduction

HOW TO READ THIS DOCUMENT

This paper explains the measures SAP strongly recommends that its customers apply to enhance the level of security with respect to certain common attack types. The sections in this paper describe, in detail, vulnerabilities and the possible exploit patterns associated with them, and how to protect applications against them. Furthermore, the paper provides guidance on how to make custom-developed applications more secure.

Be aware that this document on its own is not sufficient to ensure security and only provides guidance relating to released security notes. Implementation of security notes is a must to protect your systems. Generally, SAP recommends applying all relevant notes to all installed components, even if they are not used within a customer’s business processes. Patches related to security notes are incorporated in the next support packages issued by SAP. Several patches require additional configuration by customers after the installation of corresponding support packages.

BACKGROUND

In close collaboration with customers, partners, and external researchers, SAP has enhanced its product quality standards over the past years. A significant part of this quality enhancement process involved increasing the level of security for all major product lines for the following reasons:

- IT landscapes, including SAP® software systems, are facing more threats.
- SAP software systems are moving from back-end solutions toward open Web-based solutions.
- SAP software systems are increasingly exposed to intentional misuse.

Therefore, SAP has conducted a very thorough quality-enhancement initiative. To increase security levels, SAP utilized state-of-the-art code-scanning and penetration-testing tools for the 2010 version of SAP Business Suite software, for the existing coding of SAP Business Suite applications, and for the existing coding of the SAP NetWeaver® technology platform.

In this context, a number of corrections and improvements have been made to SAP Business Suite and the underlying SAP NetWeaver technology platform. Some of these changes may have an impact on SAP solutions as well as on customer-developed applications.

This white paper has been developed with support from SAP partner Virtual Forge GmbH. Virtual Forge is an independent security product company based in Heidelberg, Germany, and one of the leading experts in the area of SAP application security. Among other tools, SAP is using Virtual Forge’s testing software CodeProfiler in its quality assurance processes. Some of the fundamental terminology and definitions in this document have been taken, with kind permission of Virtual Forge, from the CodeProfiler report and documentation.

SUMMARY

The following table summarizes sections of this document describing vulnerabilities together with measures SAP strongly recommends that you implement. Be aware that the likelihood and impact of a vulnerability being exploited may depend on the attack scenario, how critical the supported business processes are, and which regulatory constraints apply. For example, all Internet-facing applications are generally subject to a higher risk of being attacked.
<table>
<thead>
<tr>
<th>Threat</th>
<th>Likelihood</th>
<th>Impact</th>
<th>Recommendations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transport layer security using HTTPS</td>
<td>Eavesdropping on clear text communications is well known among hackers. Conducting such attacks has become fairly easy.</td>
<td>The risk of interception of communication containing business data and user credentials (such as logon tickets, passwords, and sessions tokens)</td>
<td>Secured network communication using HTTPS from browsers to SAP® applications is key for mitigating the risk of interception of communication. Enable HTTPS at least for all communication channels that cross the network boundaries of your computer center.</td>
</tr>
<tr>
<td>Secure session handling</td>
<td>Session hijacking attacks are very common for Web-based applications.</td>
<td>Secure session handling protects against various attacks aimed at the interception of the victim’s session, which allows the attacker to view, manipulate, or delete business data.</td>
<td>SAP strongly recommends that customers implement the recommended settings, as described in this section.</td>
</tr>
<tr>
<td>Cross-site request forgery (XSRF)</td>
<td>Cross-site request forgery ranks among the most dangerous attacks.</td>
<td>An attacker makes the victim’s user agent issue requests, resulting in undesired and potentially harmful actions like viewing, manipulating, or deleting business data.</td>
<td>SAP closes vulnerabilities in standard code with the SAP Notes, from the SAP Notes tool, outlined in [1] in “Appendix 2 – References” in this paper. SAP recommends deploying the SAP Notes as soon as possible. A framework for protecting custom applications is provided.</td>
</tr>
<tr>
<td>SQL injection</td>
<td>Structured Query Language (SQL) injection is well known among hackers.</td>
<td>Exploiting dynamic SQL statements, the attacker may use malicious SQL statements to view, manipulate, and delete business data in the database.</td>
<td>SAP closes vulnerabilities in standard code with the SAP Notes outlined in [1] in “Appendix 2 – References.” SAP recommends deploying the SAP Notes as soon as possible. A framework for validating dynamic SQL statements for custom applications is also provided.</td>
</tr>
<tr>
<td>Directory traversal</td>
<td>Directory traversal is a common attack.</td>
<td>This attack exploits unintended input for programs to access directories and files on a file system level. Any data, including business data, that is accessible on the file system of the application server could be read, manipulated, or deleted without having file system access.</td>
<td>SAP closes vulnerabilities in standard code with the SAP Notes outlined in [1] in “Appendix 2 – References.” SAP recommends deploying the SAP Notes and performing additional configuration activities, as described in the notes, to activate protection measures as soon as possible. SAP also provides a framework for protecting custom applications.</td>
</tr>
<tr>
<td>Vulnerability</td>
<td>Likelihood</td>
<td>Impact</td>
<td>Recommendations</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>----------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Cross-site scripting (XSS)</td>
<td>Cross-site scripting is the most common attack on the Internet against Web applications.</td>
<td>Client-side scripts are used to access client data as well as client credentials, such as sessions, and pose a significant security risk.</td>
<td>SAP closes vulnerabilities in standard code. SAP recommends deploying the SAP Notes as soon as possible, outlined in [1] in “Appendix 2 – References.” A framework for protecting custom applications is provided.</td>
</tr>
<tr>
<td>Invoker servlet</td>
<td>This is a known vulnerability of Java application servers.</td>
<td>The invoker servlet allows unauthorized access to application servlets on Java systems.</td>
<td>The servlet must be disabled in all Java application server systems of SAP Business Suite software and SAP NetWeaver® technology platform releases.</td>
</tr>
<tr>
<td>ABAP™ code injection</td>
<td>This is a special attack requiring basic ABAP™ programming language know-how.</td>
<td>Exploiting ABAP code injection vulnerability, the attacker may do anything to view, manipulate, and delete business data.</td>
<td>SAP Notes must be implemented to help ensure that identified security vulnerabilities in standard code are closed and cannot be misused by attackers. SAP recommends deploying the SAP Notes as soon as possible, outlined in [1] in “Appendix 2 – References.”</td>
</tr>
<tr>
<td>Hard-coded user names</td>
<td>This special type of attack requires specific user names.</td>
<td>Depends on the individual vulnerability</td>
<td>SAP Notes must be implemented to help ensure that identified security vulnerabilities in standard code are closed and cannot be misused by attackers. SAP recommends deploying the SAP Notes as soon as possible, outlined in [1] in “Appendix 2 – References.”</td>
</tr>
</tbody>
</table>

**Appendix 1**
Contains an overview of vulnerabilities and corresponding countermeasures for the identified scenarios

**Appendix 2**
Links to referenced SAP Notes, SAP documentation, security guides, and so on

If you require support while implementing the SAP Notes tool referenced in this document, create an SAP customer support ticket for the primary component of the corresponding SAP Note.
Transport Layer Security Using HTTPS

HTTP protocol is used by Web-based applications running on the SAP NetWeaver Application Server (SAP NetWeaver AS) component to communicate to end-user devices (for example, a Web browser) and by communication via Web services. If HTTP is used, data and user credentials like passwords, logon tickets, and session tokens are transmitted without encryption over the network.

VULNERABILITIES

Attackers may try to actively acquire end-user credentials by sniffing the network traffic. If the communication between a user device or a remote server and the Web application is not encrypted, its confidentiality is not protected and an attacker can access not only user credentials but also all transmitted data.

COUNTERMEASURES

Encryption of HTTP traffic, especially for productive scenarios, is strongly recommended by SAP. The use of HTTPS (HTTP over SSL/TLS) helps ensure the confidentiality of the data and prevents the hijacking of sessions or security credentials (see Figure 1).

SAP recommends the following measures to implement SSL encryption:

• Use HTTPS/SSL for all access from end users’ devices to Web-based SAP applications. SSL can be terminated at infrastructure components (for example, load balancers or reverse proxies) within a server network or, if required, can be implemented from end to end between the Web browser and the application server. In addition, we recommend adding HTTPS protection to your session cookies, as described in the session-handling section of this document.

• For communication between systems within an internal network, HTTPS/SSL should be implemented if the network traffic is susceptible to sniffing.

• In addition, use encryption for server-to-server communication, especially to external services or systems. For more information on network and communication security for SAP landscapes, see [2] in “Appendix 2 – References.”

Note that valid SSL certificates that have been signed by a certification authority (CA) are required in order to be accepted by Web browsers. SAP offers corresponding services (for example, the creation of a signed server certificate for production or for testing). For more information, see [3] in “Appendix 2 – References.” In addition, SAP Notes 1527879 [4] and 1531399 [5] describe in detail how to incrementally move from HTTP to HTTPS and which steps have to be performed. After switching to encrypted network communication, corresponding applications must be tested.

Introduction of HTTPS requires a specific project within the IT organization with corresponding effort from IT and business.

Figure 1: Using HTTPS/SSL Encryption
Secure Session Handling

Stateful Web applications store the application state on the Web application server (Web AS). During communication, only the key to this state – also called session identifier or short session ID – is included with each request. In general, the session ID can be transferred as a cookie, via URL parameter, or as a hidden form field.

Besides the application state, a so-called security state (respectively, security session) might exist. A security session starts with the logon to the system and ends with the logoff from the system. SAP security session IDs will only be transmitted via nonpersistent cookies.

VULNERABILITIES

An attacker who can obtain a victim’s valid session ID can act in the attacked system on the victim’s behalf with the full set of the victim’s authorizations.

The following types of attacks can exploit session-handling vulnerabilities:

• Session hijacking is a type of attack in which the attacker steals the victim’s valid session ID and then sends a request with this session ID to the server. This can be performed, for example, by sniffing the network traffic. In some scenarios, session ID is part of the URL; this URL can be hijacked if the victim stores it in his or her bookmarks or sends it via e-mail. Assuming the session ID is still valid, the attacker can act with the full set of the victim’s authorizations.

• Session fixation is an attack in which the attacker sets the session ID for a certain user before the user authenticates with the application. This can be done by manipulating the URL that is used by the user to access the Web application. As a result, after user authentication, both the attacker and the victim know the session ID and can work on the system under the victim’s user ID.

• Session riding is a type of attack in which the attacker makes the victim’s user agent issue requests to a Web AS, resulting in undesired and potentially harmful actions. See the “Cross-Site Request Forgery (XSRF)” section for more details.

COUNTERMEASURES

It is strongly recommended that you implement the following settings on productive systems in order to improve session security.

Enable Session Security (Java)
The J2EE Engine uses the “JSESSIONID” session cookie for identifying application and security sessions. A specific protection mechanism was developed that adds an additional session identifier named “JSESSIONMARKID.” If this security mechanism is activated, the security session is identified via the additional nonpersistent cookie “JSESSIONMARKID.” “JSESSIONMARKID” changes after authentication and programmatic reauthentication, which counters session fixation and hijacking attacks. Availability of Java parameter “SessionIDRegenerationEnabled” is from SAP NetWeaver 6.40 and requires a certain support package level, which may require updating your systems as mentioned in [6] in “Appendix 2 – References.”

Enable Session Security (ABAP™ Programming Language)
Generally, an AS based on the ABAP™ programming language uses the “sap-contextid” cookie for identifying both the application session and the security session. In order to prevent session fixation and session hijacking attacks, we strongly recommend applying the following session security measures:
• Reauthentication with SAP NetWeaver 6.40, 7.00, 7.01, 7.10, and 7.11
With active reauthentication, the “sap-contextid” cookie is not enough to enter a session. Authentication credentials are checked on every round-trip. For releases 6.40 and 7.00 of SAP NetWeaver, see method in [7] in “Appendix 2 – References.” Please note that, after an upgrade to 7.01 and higher of SAP NetWeaver, release methods in [8] or [9] in “Appendix 2 – Reference” must be used. For releases 7.01, 7.10, and 7.11 of SAP NetWeaver, method [8] must be used.

• Security sessions with SAP NetWeaver 7.02, 7.20, and higher
Though the method provided with method [8] still works with SAP NetWeaver 7.02, 7.20, and higher, a new protection mechanism has been developed and should be used on newer releases of SAP NetWeaver – see method in [9] in “Appendix 2 – References.” HTTP security session management uses a new, separate cookie to identify the security session ("SAP_SESSIONID_<sid>_<client>"). A security session ID and thus the value of the "SAP_SESSIONID_<sid>_<client>" cookie changes upon authentication and programmatic reauthentication. For more information, see [9] and [10] in “Appendix 2 – References.” Before activating the HTTP security session management on an ABAP-based AS system accessed from the SAP NetWeaver Portal component, you must apply the SAP Note 1471069 [11] to the portal.

Some applications require additional configurations (for example, operating an interaction center with the SAP Customer Relationship Management application). See SAP Notes 1420203 [12], 1532777 [13], and any notes referenced in “Appendix 2 – References.” Those must be implemented before implementing the above methods.

Enable HTTPS Protection for Session Cookies
We strongly recommend using HTTPS for all browser access from end users to SAP software systems to avoid the risk of session cookies being hijacked in the network (see the “Transport Layer Security Using HTTPS” section). To prevent a browser transmitting a session cookie over an unencrypted HTTP communication channel, the cookie attribute “secure” should be set for session cookies.

For more information about how to set the attribute “SystemCookiesHTTPSProtection” for Java, see [14] and [15] in “Appendix 2 – References.” Note that those settings are available starting with version 6.40 of SAP NetWeaver and require a certain support package level as described in SAP Note 1449940 [15]. This may require updating your systems to the mentioned levels.

For ABAP systems, set parameter “login/ticket_only_by_https=1.” This parameter is available starting with version 6.10 of SAP NetWeaver AS.

Note that after enabling this attribute, plain HTTP connections will no longer work if system cookies are required to make the application work. See [5] in “Appendix 2 – References” for best practices on how to activate the recommended secure session handling.

After applying session security and HTTPS protection measures, careful regression tests must be performed for modified SAP programs and custom applications.
Cross-Site Request Forgery (XSRF)

Cross-site request forgery (also known as XSRF, CSRF, and session riding) is an attack in which an attacker is able to trick the victim into issuing an undesired request to a vulnerable application. The challenge here lies in the fact that the request might inherit the identity and privileges of the victim (automatically sent by the browser) to perform an undesired function on the victim’s behalf, like changing the victim’s e-mail address, home address, or password or performing other actions like purchasing something. XSRF is especially critical if the application is protected by a single-sign-on mechanism that does not require any user interaction (for example, the logon ticket, X.509-based authentication, Security Assertion Markup Language [SAML] with a running identity provider [IdP] session, and others) or if the user previously visited this or another application on the same application server and the session received as a result of this visit is still alive.

VULNERABILITIES

XSRF attacks generally target functions that cause a state change on the server or other critical or resource-consuming operations. Figure 2 outlines such an attack by attacker Mallory on victim Bob. The first two message exchanges between victim Bob’s user agent and the attacked Web AS of the imaginary “MyBank” serve for logging Bob on to the system and ensuring that Bob receives a valid session ID. In the following, the victim accesses a page on the right-hand side Web AS, which contains a link prepared by the attacker pointing to a vulnerable application on the “MyBank” server. If the victim is lured into clicking that link, Bob’s user agent requests the resource from the “MyBank” Web AS. Together with that request, the session ID is sent along as a cookie. Therefore, the “MyBank” Web AS accepts Bob’s request and executes the desired action of transferring €100,000 to Mallory. Bob receives in his browser the confirmation message from “MyBank” that a transfer that he had not intended has been finished successfully.

Note that more advanced mechanisms exist for making the attack less obvious, such as hiding the malicious link in an image HTML tag (<img>), using JavaScript to auto-submit form data, and so on.
COUNTERMEASURES

A common countermeasure against XSRF relies on a secret token used to ensure the “freshness” of the requests as they are received at the application server. This secret token is created after logon and stored in the user’s session. Subsequently, the token is included into state-changing local links and forms of an application. Upon receiving an HTTP request, the obtained secret token from the request can be compared with the expected secret token stored in the session. The attacker cannot forge a request reliably, since the token value for the victim is not known to him or her. There are basically two players involved in providing XSRF protection: the technology or framework (like Java Web Container, the ABAP-based Web Dynpro development environment, or binary space partitioning [BSP]) and the application built on top of it. The approach on how to protect applications depends on the characteristics of the technology. The following table gives an overview of the XSRF protection for various technologies used at SAP. In order to protect your own custom applications, you must first make the SAP framework available on the technology level by applying provided patches and, second, adapt your application to use the security framework. The table “Notes on How to Use Security Mechanisms” provides notes on how to use these security mechanisms as well as things to consider – since in some cases your applications must be adapted.

SAP closes XSRF vulnerabilities in standard code with the SAP Notes outlined in [1] in “Appendix 2 – References.” We recommend deploying the SAP Notes as soon as possible.

Notes on How to Use Security Mechanisms

<table>
<thead>
<tr>
<th>Technology</th>
<th>XSRF Protection Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Dynpro – ABAP*</td>
<td>XSRF protection within the technology -&gt; SAP Notes 1430970, 1436936, and [16] in “Appendix 2 – References”</td>
</tr>
<tr>
<td>Web Dynpro – Java</td>
<td>XSRF protection within the technology -&gt; SAP Notes 1521024, 1327872</td>
</tr>
<tr>
<td>SAP NetWeaver® Application Server Java</td>
<td>XSRF protection based on a technology API that is adopted by the applications; see [17] in “Appendix 2 – References”</td>
</tr>
<tr>
<td>BSP applications</td>
<td>XSRF protection within the framework -&gt; SAP Note 1458171</td>
</tr>
<tr>
<td>ITS services</td>
<td>XSRF protection within the framework -&gt; SAP Note 1481392</td>
</tr>
</tbody>
</table>
SQL Injection

SAP applications as well as custom-developed applications rely on relational database management system (RDBMS) servers. The information is stored and retrieved with structured query language (SQL) statements. The vulnerability for ABAP-based implementations lies in the creation of dynamic SQL statements within program code (using native or open SQL), which allows user input to be executed directly without filtering or verification.

VULNERABILITIES

Attackers are successful if they are able to change the semantics of a dynamic SQL statement for their benefit or are able to insert their own statements into the application. Figure 3 shows how malicious user input can lead to data leakage: a "where" clause is dynamically built upon user input, which retrieves unauthorized database content (here for open SQL).

The programmer expects single values in a string-named input that the program receives. As long as input contains only character strings like "LH," the program works as intended. An attacker could put a string like "'LH' OR CARRID LIKE '%,'" which in this example selects all entries from the database table.

User input can come directly from an HTML form within a Web application, a URL, an input field in any SAP user interface, or other inputs (for example, within remote function calls from other systems or data-loading activities).

An attacker can exploit this vulnerability to execute arbitrary database commands to retrieve, modify, or remove data persisted by the system. For example, an attacker could gain unauthorized access to critical data like credit card numbers or manipulate the outcome of a business process by manipulating the data read – see [18] in "Appendix 2 – References."

COUNTERMEASURES

Open SQL for ABAP already provides some implicit protection against SQL code injection, and SAP further improved the quality of code in order to prevent SQL injection attacks on SAP products. Implement the provided SAP Notes in order to avoid SQL injection vulnerabilities for SAP products and applications – see [1] in "Appendix 2 – References." Please consider that once the patches have been applied, SAP applications will not accept arbitrary input for dynamic SQL statements. This is especially important if your own applications perform calls to SAP applications that are affected by the SAP Notes. Please test your corresponding applications and adapt them if needed.

Further, if you have modified SAP applications or created your own programs that involve dynamic SQL statements (native or open SQL), consider improving your own code quality (for example, by replacing dynamic code with static code as far as possible). Furthermore, SAP recommends using open SQL instead of native SQL. For all releases, SAP provides standard methods that can help to avoid SQL injections (for example, checks for dynamic SQL statements). For more information, see the SAP Note 1520356 [19] in "Appendix 2 – References."

Figure 3: Example of SQL Injection

<table>
<thead>
<tr>
<th>Original SQL statement</th>
<th>Example of an SQL injection attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT * FROM spfl INTO TABLE itab WHERE CARRID = '&amp;&amp; userinput &amp;&amp;'</td>
<td>SELECT * FROM spfl INTO TABLE itab WHERE CARRID = 'LH' OR CARRID LIKE '%,'</td>
</tr>
</tbody>
</table>
Directory Traversal

Directory traversal attacks in ABAP applications work by manipulating file name or path information by feeding special characters into a string that represents a file locator. When such a string is used to access a file, an application can be tricked into opening files to which the user should not have access.

This attack works because the application fails to detect and remove the command characters in the input that is used as part of the file locator. It affects files in all directories that the vulnerable application has access to. This may include files within the company network.

VULNERABILITIES

Many applications access files on the server in order to read or write data. Common-use cases include temporary persistence of file uploads, access to configuration files, and access to templates. On the operating system level, files are identified by file locators. These file locators contain information about the drive or file share, directory, name, and extension of a given file. There are instances in which part of the information of such a file locator is external input. For example, the name of a file uploaded to the server may also be used to store it in a temporary folder. But external input might contain special characters that could be used to tamper with the overall file locator. This vulnerability affects improper use of the ABAP commands OPEN DATASET, READ DATASET, DELETE DATASET, and TRANSFER.

Such vulnerabilities can be exploited in this way: A direct exploit is possible when an attacker gains direct access to critical business assets. These could be the database or any documents with business or confidential data. If there are no business assets on the vulnerable server, the attacker can still exploit the vulnerability indirectly by accessing files that hold important technical information, such as log or configuration files. Such files may reveal information that supports further attacks. In any case, unauthorized remote access to arbitrary files on a server is a security risk that requires immediate attention.

By manipulating which files an application will access, at least the following attacks are possible:
• Read and write access to critical configuration files. This will help attackers to further penetrate the attacked system.
• Read and write access to log files
• Read and write access to the file persistence of a productive database (if the vulnerability allows access to the database server)
• Read and write access to confidential or secret files

Some of those risks may violate regulatory compliance, as this vulnerability allows for unprivileged access to critical business data – see [20] in “Appendix 2 – References.”

COUNTERMEASURES

To protect your computer systems against directory traversal, SAP offers a security service for mapping physical paths and file names to logical paths or file names. This special mechanism allows you to minimize or completely eliminate system vulnerability. When applying the corresponding updates or correction instructions [21], new logical file names and new logical paths will be created in the system. These logical file names and file paths will not contain any assignment to physical file names or file paths. As long as the customer’s system administrator does not maintain these physical file names, the program’s file name validation will not be performed. To prepare for the fix, you should do an analysis of your application in order to find out where access to the file system is necessary. Identify which programs perform the access. More details can be found in the SAP Note 1497003 [21], the SAP NetWeaver Security Guide [22], or the SAP Secure Programming Guide [23] in “Appendix 2 – References.”

SAP closes directory traversal vulnerabilities in standard code with the SAP Notes outlined in [1] in “Appendix 2 – References.” SAP recommends deploying the notes as soon as possible.
Cross-Site Scripting

Cross-site scripting (XSS) attacks work by inserting special command sequences into an HTML page via (user) input. When the HTML page is loaded, the planted code will execute in a user’s browser. This attack works because the Web application fails to detect and encode the attack code in any given input and affects all users opening the vulnerable page.

VULNERABILITIES

By means of cross-site scripting, an attacker can manipulate HTML pages (that are not his or her own) that someone else will read in a browser later. While most other attacks target Web servers, XSS is an attack against clients. In other words, if developers don’t code Web applications securely, they jeopardize the safety of their user’s data through XSS attacks.

There are two main variants of XSS attacks:

- **Reflected XSS** (Figure 4): In an immediate server-side input reflection, parameters passed to a Web page reappear in its HTML source. An attacker lures users with a specially crafted link to a vulnerable Web application with an e-mail, a message board post, or other communication.

- **Stored XSS** (Figure 5): The malicious command sequence provided by an attacker is stored on the database of the Web application and is permanently displayed to other users in the course of regular browsing. It does not require any specially crafted links for execution.
It is important to note that “input” is not limited to data that users actually type into input fields of a Web page. Instead, all parameter values submitted to a Web page must be considered as input – see [24] in “Appendix 2 – References.”

By embedding HTML into a page that someone else will open, at least the following attacks are possible:
- Identity theft: stealing and continuing another user’s session
- Stealing any form input (credit card data, credentials, and so on) by submitting the content to a hacker-controlled server instead of the real destination
- Remotely accessing the client’s file system through embedded ActiveX controls
- Defacing a Web site by including fake content
- Intranet port scanning
- Key logging
- Freezing any client system through endless loops or pop-ups

COUNTERMEASURES

To prevent XSS vulnerability of a Web application, it is important that all user input is encoded before output. The encoding must be done so that the browser recognizes it as text only and cannot be misinterpreted as active content (like JavaScript commands). SAP offers not just one encoding function but a selection of functions for various contexts. It is always important to use the appropriate encoding function for the protection of a Web application against XSS attacks – see [25] and [26] in “Appendix 2 – References.”

SAP improved the quality of SAP coding by fixing such issues in recent support packages and by providing the SAP Notes outlined in [1] in “Appendix 2 – References.” SAP recommends implementing the notes as soon as possible.
Invoker Servlet

The invoker servlet is intended only to be used for rapid prototyping and allows HTTP clients to invoke servlets that have not been declared in the application’s /WEB-INF/web.xml file. Using the invoker servlet, it is possible to call arbitrary servlets by servlet name (as declared in the <servlet-name> tag of the web.xml file) or by its fully qualified servlet class name. In this case, it is not necessary to declare the servlet in the web.xml file. All HTTP requests of the form /<context-root>/servlet/<servlet-name-or-class> get direct access to this servlet named <servlet-name-or-class>. For more information about invoker servlet, see [27] in “Appendix 2 – References.”

VULNERABILITY

A specially crafted URL using the invoker servlet feature can allow unauthenticated access to arbitrary servlets. In addition, there is no authentication needed in order to invoke these servlets.

COUNTERMEASURES

The invoker servlet feature has to be disabled to close the security gap described here. Do the following:

1. Update to the latest patch level that corresponds to your support package and release – see SAP Note 1445998 [28] in “Appendix 2 – References.”
2. Identify whether any requested scenarios rely on the invoker servlet.
3. Disable the vulnerable feature by changing the value of the “EnableInvokerServletGlobally” property of the servlet_jsp service on the server nodes to “false” – see SAP Note 1445998 [28].
4. For adaptation of existing applications, which use the invoker servlet feature, see the PDF document attached to SAP Note 1445998 [28].
5. If you are using SAP NetWeaver Portal, see SAP Note 1467771.

The invoker servlet has been disabled by default in SAP NetWeaver 7.20 (in the patch levels described in Note [28]) and in the initial shipment of SAP NetWeaver 7.30.
ABAP Code Injection

The ABAP command generate subroutine pool/insert report is used to dynamically construct an ABAP program or ABAP report. This is done by appending strings that are usually read from a data source to an internal table. Once the ABAP program has been assembled, it can be executed (for example, with the command perform).

VULNERABILITIES

Such coding practices are very dangerous, as they may allow construction of malicious code on the fly and leave no traces of this code in the system.

If a user can execute arbitrary ABAP commands on an SAP software system, then the system must be considered to be completely compromised:

• Read and write access to all data in the database
• Execution of arbitrary business logic
• Violation of regulatory compliance

COUNTERMEASURES

SAP closes vulnerabilities in standard code with the SAP Notes outlined in [1] in “Appendix 2 – References.” SAP recommends deploying the notes as soon as possible.

For custom code, make sure that no external input is used as part of a dynamically generated ABAP program or ABAP report. If you must use external input, discuss this with your security department and make sure you have very strict white list input filters in place – see [29] in “Appendix 2 – References.”
Hard-Coded User Names

In ABAP, the system field SY-UNAME holds the name of the currently logged-on user. Often developers build test code that is executed only when they are logged on to the system. If such code gets transported to productive systems, a user with the same SY-UNAME will execute this test code accidentally in the productive system.

VULNERABILITIES

Attackers might use this technique to install back doors into applications by adding hidden functions that will only be executed for a specific logged-on user. This is especially dangerous if such applications can be accessed remotely.

COUNTERMEASURES

Do not write code that is activated based on the SY-UNAME of the currently logged-on user – see [30] in “Appendix 2 – References.”

Remove all instances of hard-coded user names in your code in productive systems. SAP has improved the quality of our coding by fixing such issues in recent support packages and by providing the SAP Notes outlined in [1] in “Appendix 2 – References.”
Appendix 1 – Overview Table

The table shows the vulnerabilities, affected applications, and scenarios (for example, “Web-based” signifies that Web-based applications are in scope). In addition, the table provides a summary of recommended countermeasures and impact that the customer may experience (for example, additional effort to perform testing).

<table>
<thead>
<tr>
<th>Vulnerability</th>
<th>Affected Scenarios</th>
<th>Countermeasures (See [31] in “Appendix 2 – References.”)</th>
<th>Impact by Applying Countermeasure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Session handling</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Session fixation</td>
<td>Web-based SAP* and custom applications (Java and ABAP™ programming languages)</td>
<td>Configure session ID regeneration (Java); configure reauthentication and security session handling (ABAP).</td>
<td>Applications must be tested.</td>
</tr>
<tr>
<td>Session hijacking</td>
<td>Web-based SAP and custom applications (Java and ABAP)</td>
<td>See measures for session fixation. Implement HTTPS/SSL and HTTPS session protection.</td>
<td>Applications must be tested. Establishing HTTPS encryption may lead to higher effort. Nonencrypted HTTP access to applications may not work.</td>
</tr>
<tr>
<td><strong>Cross-site request forgery (XSRF)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XSRF: Web Dynpro</td>
<td>Web Dynpro SAP and custom applications (ABAP and Java)</td>
<td>Apply patches for the SAP NetWeaver* technology platform. Review and adapt endangered custom code.</td>
<td>Applications must be tested.</td>
</tr>
<tr>
<td>XSRF: SAP applications</td>
<td>Web-based SAP applications (Java and ABAP [BSP, ITS])</td>
<td>Apply ABAP and Java patches for SAP NetWeaver and SAP applications.</td>
<td>Applications must be tested.</td>
</tr>
<tr>
<td>XSRF: custom applications</td>
<td>Web-based custom applications (Java and ABAP [BSP, ITS])</td>
<td>Review and adapt endangered custom code to use SAP framework.</td>
<td>Reprogramming and testing custom code may lead to significantly higher effort.</td>
</tr>
<tr>
<td><strong>Structured query language (SQL) injection</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SQL injection: SAP applications</td>
<td>SAP applications (ABAP)</td>
<td>Apply SAP corrections.</td>
<td>Applications must be tested.</td>
</tr>
<tr>
<td>SQL injection: custom applications</td>
<td>Custom applications (ABAP)</td>
<td>Apply SAP corrections described in the SAP Note 1520356, and review and adapt custom applications to use SAP framework.</td>
<td>Reprogramming and testing custom code may lead to significantly higher effort.</td>
</tr>
<tr>
<td>Vulnerability</td>
<td>Affected Scenarios</td>
<td>Countermeasures (See [31] in “Appendix 2 – References.”)</td>
<td>Impact by Applying Countermeasure</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Directory traversal</td>
<td>SAP applications (ABAP)</td>
<td>Apply SAP corrections described in the SAP Note 1497003. Physical file names must be maintained for logical file names and paths to enable the validation feature, as described in the application-specific notes in [1] in “Appendix 2 – References.”</td>
<td>Applications must be tested.</td>
</tr>
<tr>
<td>Directory traversal:</td>
<td>Custom applications (ABAP)</td>
<td>Apply corrections in the SAP Note 1497003, and review and reprogram custom code using SAP-provided framework.</td>
<td>Reprogramming and testing custom code may lead to significantly higher effort.</td>
</tr>
<tr>
<td>custom applications</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XSS (cross-site</td>
<td>Web-based SAP and custom applications (ABAP and Java)</td>
<td>Apply SAP corrections. Review and adapt endangered custom code.</td>
<td>Applications must be tested.</td>
</tr>
<tr>
<td>scripting)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Invoker servlet</td>
<td>Web-based custom applications (Java)</td>
<td>Apply patches and disable invoker servlet; review and adapt applications relying on it.</td>
<td>If custom application relying on invoker servlet is not adapted, it will not work after disabling invoker servlet.</td>
</tr>
<tr>
<td>ABAP code injection</td>
<td>SAP applications (ABAP)</td>
<td>Apply SAP corrections.</td>
<td>Applications must be tested.</td>
</tr>
<tr>
<td>Hard-coded user names</td>
<td>SAP applications (ABAP)</td>
<td>Apply SAP corrections.</td>
<td>Applications must be tested.</td>
</tr>
</tbody>
</table>


Appendix 2 – References

https://service.sap.com/sap/support/notes/1533030

http://help.sap.com/saphelp_nw70ehp2/helpdata/en/f/7d5b5386f64b5585e10000009b38f8cf/frameset.htm


https://service.sap.com/sap/support/notes/1527879

https://service.sap.com/sap/support/notes/1531399

https://service.sap.com/sap/support/notes/1310561

[7] SAP Note 1266780 – User Check for Each HTTP Request
https://service.sap.com/sap/support/notes/1266780

[8] SAP Note 1277022 – Security Note: Session Forwarding by URL
https://service.sap.com/sap/support/notes/1277022

https://service.sap.com/sap/support/notes/1322944

[10] Activating HTTP Security Session Management on AS ABAP

https://service.sap.com/sap/support/notes/1471069

[12] Note 1420203 – Enable Foreign Access to a Stateful HTTP Session
https://service.sap.com/sap/support/notes/1420203

[13] SAP Note 1532777 – Collective Note: ABAP Session Protection Recommendations
https://service.sap.com/sap/support/notes/1532777

[14] HTTP Provider Service
http://help.sap.com/saphelp_nw70/helpdata/en/52/46f6a089754e3a964a5d932eb9e9b8b/content.htm

[15] SAP Note 1449940 – Browsing Web Shop via HTTP with SessionIDRegenerationEnabled
https://service.sap.com/sap/support/notes/1449940

[16] Web Dynpro ABAP
http://help.sap.com/saphelp_nw70ehp2/helpdata/en/f4/b03e8cbd28414bbf73b0f10d3b7890/frameset.htm

[17] SAP Note 1450166 – Unauthorized Usage of Application Functionality
https://service.sap.com/sap/support/notes/1450166

[18] Source of this text element: Virtual Forge CodeProfiler documentation.

[19] SAP Note 1520356 – Avoiding SQL Injections: Partner Documentation
https://service.sap.com/sap/support/notes/1520356


[21] SAP Note 1497003 – Potential Directory Traversals in Applications
https://service.sap.com/sap/support/notes/1497003

[22] SAP NetWeaver Security Guide
http://help.sap.com/saphelp_nw70ehp2/helpdata/en/A8/F8BC0DD5DB41939A45D4FB95FC9D34/frameset.htm

[23] Secure Programming Guide


http://help.sap.com/saphelp_nw70ehp2/helpdata/en/84/6ad7dbe5254ddee10000000a1553f7/frameset.htm

[26] SAPinsider Article, “Guard Your Web Applications Against XSS Attacks”
www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/80e7b8ca-37cc-2910-a184-974e95dd1e06

[27] Invoker Servlet

[28] Note 1445998 – Disabling Invoker Servlet
https://service.sap.com/sap/support/notes/1445998

[29] Source of this text element: Virtual Forge CodeProfiler documentation.


[31] Be aware that patches SAP provides on this patch day will be incorporated into next SAP support packages automatically.

This white paper has been developed with kind support from: