




 Secrets


 ABAP


 Apex


 C


 C++


 CloudFormation


 COBOL


 C#


 CSS


 Flex


 Go


 HTML


 **Java**


 JavaScript


 Kotlin


 Objective C


 PHP


 PL/I


 PL/SQL


 Python


 RPG


 Ruby


 Scala


 Swift


 Terraform


 Text


 TypeScript

 T-SQL

 VB.NET

 VB6

 XML



## Java static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your JAVA code

All rules632

Vulnerability53

Bug154

Security Hotspot36


Code Smell389

Quick Fix42


Tags ▾

Search by name... 🔍


vulnerable to injection attacks

 Vulnerability


I/O function calls should not be vulnerable to path injection attacks

 Vulnerability


LDAP queries should not be vulnerable to injection attacks

 Vulnerability


OS commands should not be vulnerable to command injection attacks

 Vulnerability


"@SpringBootApplication" and "@ComponentScan" should not be used in the default package

 Bug


"@Controller" classes that use "@SessionAttributes" must call "setComplete" on their "SessionStatus" objects

 Bug


"wait" should not be called when multiple locks are held

 Bug


"PreparedStatement" and "ResultSet" methods should be called with valid indices

 Bug


Files opened in append mode should not be used with ObjectOutputStream

 Bug

"wait(...)" should be used instead of "Thread.sleep(...)" when a lock is held

 Bug

Printf-style format strings should not lead to unexpected behavior at runtime

 Bug

### Endpoints should not be vulnerable to reflected cross-site scripting (XSS) attacks

Analyze your code

Vulnerability

Blocker

injection cwe sans-top25 owasp

User-provided data, such as URL parameters, POST data payloads, or cookies, should always be considered untrusted and tainted. Furthermore, when processing an HTTP request, a web server may copy user-provided data into the body of the HTTP response that is sent back to the user. This behavior is called a "reflection". Endpoints reflecting tainted data could allow attackers to inject code that would eventually be executed in the user's browser. This could enable a wide range of serious attacks like accessing/modifying sensitive information or impersonating other users.

Typically, the solution is one of the following:

- Validate user-provided data based on a whitelist and reject input that is not allowed.
- Sanitize user-provided data from any characters that can be used for malicious purposes.
- Encode user-provided data when it is reflected back in the HTTP response. Adjust the encoding to the output context so that, for example, HTML encoding is used for HTML content, HTML attribute encoding is used for attribute values, and JavaScript encoding is used for server-generated JavaScript.

When sanitizing or encoding data, it is recommended to only use libraries specifically designed for security purposes. Also, make sure that the library you are using is being actively maintained and is kept up-to-date with the latest discovered vulnerabilities.

#### Noncompliant Code Example

```
protected void doGet(HttpServletRequest req, HttpServletResponse res) {
    String name = req.getParameter("name");
    PrintWriter out = res.getWriter();
    out.write("Hello " + name); // Noncompliant
}
```

#### Compliant Solution






```
protected void doGet(HttpServletRequest req, HttpServletResponse res) {
    String name = req.getParameter("name");
    String encodedName = org.owasp.encoder.Encode.forHtml(name);
    PrintWriter out = res.getWriter();
    out.write("Hello " + encodedName);
}
```

#### See

- [OWASP Top 10 2021 Category A3](#) - Injection
- [OWASP Cheat Sheet](#) - XSS Prevention Cheat Sheet
- [OWASP Top 10 2017 Category A7](#) - Cross-Site Scripting (XSS)
- [MITRE, CWE-79](#) - Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

https://rules.sonarsource.com/java/RSPEC-5131

1/2


<b>Methods "wait(...)", "notify()" and "notifyAll()" should not be called on Thread instances</b>  Bug
<b>Methods should not call same-class methods with incompatible "@Transactional" values</b>  Bug
<b>Recursion should not be infinite</b>  Bug
<b>Loops should not be infinite</b>  Bug

- [SANS Top 25](#) - Insecure Interaction Between Components

Available In:  
**sonarcloud**  | **sonarqube**  Developer Edition

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.  
[Privacy Policy](#)