




 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... 🔍

Redundant constructors/methods should be avoided in records

Code Smell

Records should be used instead of ordinary classes when representing immutable data structure

Code Smell

"Stream.toList()" method should be used instead of "collectors" when unmodifiable list needed

Code Smell

Operator "instanceof" should be used instead of "A.class.isInstance()"

Code Smell

String multiline concatenation should be replaced with Text Blocks

Code Smell

Single-character alternations in regular expressions should be replaced with character classes

Code Smell

Reluctant quantifiers in regular expressions should be followed by an expression that can't match the empty string

Code Smell

Constructors of an "abstract" class should not be declared "public"

Code Smell

Similar tests should be grouped in a single Parameterized test

Code Smell

Tests should be stable

Code Smell

Test methods should not contain too many assertions

Code Smell

XML signatures should be validated securely

Analyze your code

Vulnerability

Major

XML document can be signed to ensure data integrity and authentication. The signature should be verified and validated to make sure it's secure. For instance, signatures based on weak cipher algorithms like MD5 should be rejected, and an XML document should not contain hostile constructs that can lead to Denial of Services attacks, like a large number of SignedInfo elements.

Noncompliant Code Example

The Java XML Digital Signature API doesn't use a strong signature validation mode by default:

```
DOMValidateContext valContext = new DOMValidateContext(new K
```

Compliant Solution

The Java XML Digital Signature API offers a secure validation mode to protect against various [security issues](#):

```
DOMValidateContext valContext = new DOMValidateContext(new K  
valContext.setProperty("org.jcp.xml.dsig.secureValidation",
```

See

- [Oracle Java Documentation](#) - XML Digital Signature API Overview and Tutorial
- [OWASP Top 10 2017 Category A3](#) - Sensitive Data Exposure
- [MITRE, CWE-347](#) - Improper Verification of Cryptographic Signature

Available In:

sonarlint




sonarcloud

sonarqube

© 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](#)

1/2

https://rules.sonarsource.com/java/RSPEC-6377

<div>AssertJ "assertThatThrownBy" should not be used alone</div> <div> Code Smell</div>
<div>Character classes in regular expressions should not contain the same character twice</div> <div> Code Smell</div>
<div>Names of regular expressions named groups should be used</div> <div> Code Smell</div>
<div>Regexes containing characters subject to normalization should use the CANON_EQ flag</div>