




 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 rules 632

Vulnerability 53

Bug 154

Security Hotspot 36

Code Smell 389

Quick Fix 42

Tags ▾

Search by name... 🔍

algorithms is security-sensitive

Security Hotspot

Using pseudorandom number generators (PRNGs) is security-sensitive

Security Hotspot

Mocking all non-private methods of a class should be avoided

Code Smell

Empty lines should not be tested with regex MULTILINE flag

Code Smell

Methods setUp() and tearDown() should be correctly annotated starting with JUnit4

Code Smell

Class members annotated with "@VisibleForTesting" should not be accessed from production code

Code Smell

"String#replace" should be preferred to "String#replaceAll"

Code Smell

Derived exceptions should not hide their parents' catch blocks

Code Smell

String offset-based methods should be preferred for finding substrings from offsets

Code Smell

"default" clauses should be last

Code Smell

"equals" method parameters should not be marked "@Nonnull"

Code Smell

A conditionally executed single line

Weak SSL/TLS protocols should not be used

Analyze your code

Vulnerability Critical ? cwe privacy owasp sans-top25

This rule raises an issue when an insecure TLS protocol version (i.e. a protocol different from "TLSv1.2", "TLSv1.3", "DTLSv1.2", or "DTLSv1.3") is used or allowed.

It is recommended to enforce TLS 1.2 as the minimum protocol version and to disallow older versions like TLS 1.0. Failure to do so could open the door to downgrade attacks: a malicious actor who is able to intercept the connection could modify the requested protocol version and downgrade it to a less secure version.

Noncompliant Code Example

javax.net.ssl.SSLContext library:

context = SSLContext.getInstance("TLSv1.1"); // Noncompliant

okhttp library:

ConnectionSpec spec = new ConnectionSpec.Builder(ConnectionSpec.
.tlsVersions(TlsVersion.TLS_1_1) // Noncompliant
.build());

Compliant Solution

javax.net.ssl.SSLContext library:

context = SSLContext.getInstance("TLSv1.2"); // Compliant

okhttp library:

ConnectionSpec spec = new ConnectionSpec.Builder(ConnectionSpec.
.tlsVersions(TlsVersion.TLS_1_2) // Compliant
.build());

See





- OWASP Top 10 2021 Category A2 - Cryptographic Failures
- OWASP Top 10 2021 Category A7 - Identification and Authentication Failures
- OWASP Top 10 2017 Category A3 - Sensitive Data Exposure
- OWASP Top 10 2017 Category A6 - Security Misconfiguration
- MITRE, CWE-327 - Inadequate Encryption Strength
- MITRE, CWE-326 - Use of a Broken or Risky Cryptographic Algorithm
- SANS Top 25 - Porous Defenses
- Diagnosing TLS, SSL, and HTTPS
- SSL and TLS Deployment Best Practices - Use secure protocols

Available In:

sonarlint | sonarcloud | sonarqube

https://rules.sonarsource.com/java/RSPEC-4423

1/2

<div>if conditionally executed single line should be denoted by indentation</div> <div> Code Smell</div>	
<div>Conditionals should start on new lines</div> <div> Code Smell</div>	
<div>Cognitive Complexity of methods should not be too high</div> <div> Code Smell</div>	
<div>Factory method injection should be used in "@Configuration" classes</div> <div> Code Smell</div>	
<div>"static" base class members should not be accessed via derived types</div> <div></div>	

© 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](#)