




 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... 🔍


Loops with at most one iteration should be refactored

 Bug


Variables should not be self-assigned

 Bug


"StringBuilder" and "StringBuffer" should not be instantiated with a character

 Bug


Methods should not be named "toString", "hashCode" or "equal"

 Bug


"Thread.run()" should not be called directly

 Bug


"equals" method overrides should accept "Object" parameters

 Bug


The Object.finalize() method should not be called

 Bug


Enabling file access for WebViews is security-sensitive

 Security Hotspot


Enabling JavaScript support for WebViews is security-sensitive

 Security Hotspot

Constructing arguments of system commands from user input is security-sensitive


 Security Hotspot


Using unencrypted files in mobile applications is security-sensitive


 Security Hotspot

Package declaration should match source file directory

Analyze your code

 Code Smell

 Critical




 pitfall

By convention, a Java class' physical location (source directories) and its logical representation (packages) should be kept in sync. Thus a Java file located at "src/org/bar/Foo.java" should have a package of "org.bar".

Unfortunately, this convention is not enforced by Java compilers, and nothing prevents a developer from making the "Foo.java" class part of the "com.apple" package, which could degrade the maintainability of both the class and its application.

Similarly, source placed in a folder with dots in its name instead of having the equivalent folder structure will compile but cause problems at run time. For instance, code with a package declaration of org.foo.bar that is placed in org/foo.bar will compile, but the classloader will always search for the class into the folder based on package structure, and will consequently expect sources to be in org/foo/bar folder. foo.bar is therefore not a proper folder name for sources.




Available In:

 |  | 

© 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](#)

https://rules.sonarsource.com/java/RSPEC-1598

1/2

<p>Using biometric authentication without a cryptographic solution is security-sensitive</p> <p> Security Hotspot</p>
<p>Using unencrypted databases in mobile applications is security-sensitive</p> <p> Security Hotspot</p>
<p>Authorizing non-authenticated users to use keys in the Android KeyStore is security-sensitive</p> <p> Security Hotspot</p>
<p>Allowing user enumeration is security-sensitive</p>