




 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... 🔍


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


Using biometric authentication without a cryptographic solution is security-sensitive


 Security Hotspot


Using unencrypted databases in

Generic wildcard types should not be used in return types

Analyze your code

 Code Smell

 Critical

 pitfall

It is highly recommended **not** to use wildcard types as return types. Because the type inference rules are fairly complex it is unlikely the user of that API will know how to use it correctly.

Let's take the example of method returning a "List<? extends Animal>". Is it possible on this list to add a Dog, a Cat, ... we simply don't know. And neither does the compiler, which is why it will not allow such a direct use. The use of wildcard types should be limited to method parameters.

This rule raises an issue when a method returns a wildcard type.

Noncompliant Code Example

List<? extends Animal> getAnimals(){...}

Compliant Solution

List<Animal> getAnimals(){...}

or

List<Dog> getAnimals(){...}

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](#)

https://rules.sonarsource.com/java/RSPEC-1452

1/2

<div>mobile applications is security-sensitive</div> <div> Security Hotspot</div>
<div>Authorizing non-authenticated users to use keys in the Android KeyStore is security-sensitive</div> <div> Security Hotspot</div>
<div>Allowing user enumeration is security-sensitive</div> <div> Security Hotspot</div>
<div>Allowing requests with excessive content length is security-sensitive</div> <div> Security Hotspot</div>