




 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... 🔍


 Vulnerability


Persistent entities should not be used as arguments of "@RequestMapping" methods


 Vulnerability


 "HttpSecurity" URL patterns should be correctly ordered


 Vulnerability


 LDAP connections should be authenticated


 Vulnerability


 Cryptographic keys should be robust


 Vulnerability


 Weak SSL/TLS protocols should not be used


 Vulnerability


 "SecureRandom" seeds should not be predictable


 Vulnerability


 Cipher Block Chaining IVs should be unpredictable


 Vulnerability


 Basic authentication should not be used


 Vulnerability


 Regular expressions should not be vulnerable to Denial of Service attacks

 Vulnerability




 "HttpServletRequest.getRequestSession" should not be used

 Vulnerability

 Hashes should include an unpredictable salt

 Vulnerability

Methods returns should not be invariant

 Code Smell  Blocker 




When a method is designed to return an invariant value, it may be poor design, but it shouldn't adversely affect the outcome of your program. However, when it happens on all paths through the logic, it is surely a bug.

This rule raises an issue when a method contains several `return` statements that all return the same value.

Noncompliant Code Example

```
int foo(int a) {
    int b = 12;
    if (a == 1) {
        return b;
    }
    return b; // Noncompliant
}
```

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-3516

1/2

<p>Calls to methods should not trigger an <code>IllegalArgumentException</code></p> <p> Bug</p>
<p>Unsupported methods should not be called on some collection implementations</p> <p> Bug</p>
<p>Cast operations should not trigger a <code>ClassCastException</code></p> <p> Bug</p>
<p>Members ignored during record serialization should not be used</p> <p> Bug</p>