


-  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... 🔍

 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


Calls to methods should not trigger an IllegalArgumentException

 Bug

Unsupported methods should not be called on some collection implementations


 Bug


Cast operations should not trigger a ClassCastException


 Bug

"ThreadGroup" should not be used

Analyze your code

 Code Smell

 Blocker

 design cert suspicious

There is little valid reason to use the methods of the ThreadGroup class. Some are deprecated (allowThreadSuspension(), resume(), stop(), and suspend()), some are obsolete, others aren't thread-safe, and still others are insecure (activeCount(), enumerate()). For these reasons, any use of ThreadGroup is suspicious and should be avoided.

Compliant Solution

```
ThreadFactory threadFactory = Executors.defaultThreadFactory();
ThreadPoolExecutor executorPool = new ThreadPoolExecutor(3,

for (int i = 0; i < 10; i++) {
    executorPool.execute(new JobThread("Job: " + i));
}

System.out.println(executorPool.getActiveCount()); // Compliant
executorPool.shutdown();
```

See

- [CERT, TH101-J](#). - Do not invoke ThreadGroup methods

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-3014

1/2

<p>Members ignored during record serialization should not be used</p> <p> Bug</p>
<p>Map "computeIfAbsent()" and "computeIfPresent()" should not be used to add "null" values.</p> <p> Bug</p>
<p>Regex lookahead assertions should not be contradictory</p> <p> Bug</p>
<p>Back references in regular expressions should only refer to capturing groups that are matched before the reference</p>