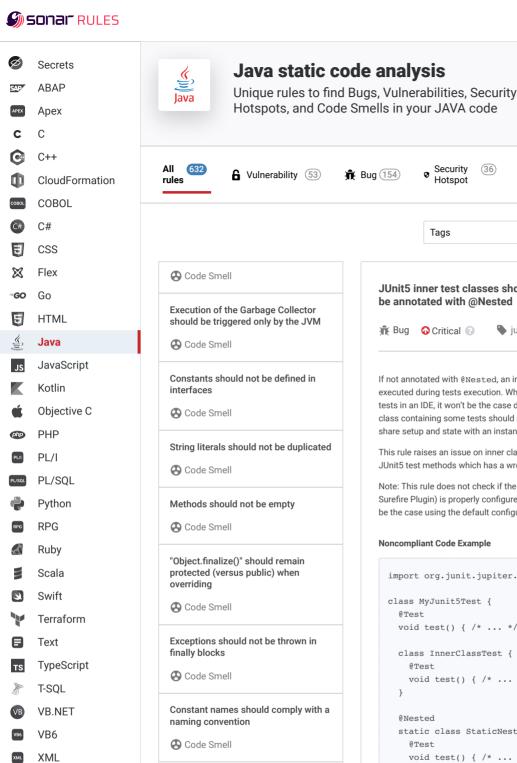
Products ✓

O Quick 42 Fix



The Object.finalize() method should

XML operations should not be

vulnerable to injection attacks

JSON operations should not be

XML signatures should be validated

vulnerable to injection attacks

@Test

@Test

@Test

void test() { /\* ... \*/ }

void test() { /\* ... \*/ }

static class StaticNestedClassTest {

class InnerClassTest {

not be overridden

Code Smell

Vulnerability

Vulnerability

Vulnerability

securely

```
Q
                                           Search by name.
JUnit5 inner test classes should
                                                 Analyze your code
be annotated with @Nested
                            junit tests
If not annotated with @Nested, an inner class containing some tests will never be
executed during tests execution. While you could still be able to manually run its
tests in an IDE, it won't be the case during the build. By contrast, a static nested
class containing some tests should not be annotated with @Nested, JUnit5 will not
share setup and state with an instance of its enclosing class.
This rule raises an issue on inner classes and static nested classes containing
JUnit5 test methods which has a wrong usage of @Nested annotation.
Note: This rule does not check if the context in which JUnit 5 is running (e.g. Maven
Surefire Plugin) is properly configured to execute static nested classes, it could not
be the case using the default configuration.
  import org.junit.jupiter.api.Test;
    void test() { /* ... */ }
    class InnerClassTest { // Noncompliant, missing @Nested an
       void test() { /* ... */ }
    static class StaticNestedClassTest { // Noncompliant, inva
       void test() { /* ... */ }
Compliant Solution
  import org.junit.jupiter.api.Test;
  import org.junit.jupiter.api.Nested;
  class MyJunit5Test {
```

(389)

sava static code anary.	310
XML parsers should not be vulnerable to Denial of Service attacks	ļ
♠ Vulnerability	
XML parsers should not load external schemas	
★ Vulnerability	
Mobile database encryption keys should not be disclosed	
ெ Vulnerability	
Reflection should not be vulnerable to	)
injection attacks	
★ Vulnerability	

```
void test() { /* ... */ }
}

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