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...

Exceptions should not be thrown in finally blocks
Code Smell

Constant names should comply with a naming convention
Code Smell

The Object.finalize() method should not be overridden
Code Smell

XML operations should not be vulnerable to injection attacks
Vulnerability

JSON operations should not be vulnerable to injection attacks
Vulnerability

XML signatures should be validated securely
Vulnerability

XML parsers should not be vulnerable to Denial of Service attacks
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

Authorizations should be based on strong decisions
Vulnerability

Getters and setters should access the expected fields

Analyze your code

Bug | Critical | pitfall

Getters and setters provide a way to enforce encapsulation by providing `public` methods that give controlled access to `private` fields. However in classes with multiple fields it is not unusual that copy and paste is used to quickly create the needed getters and setters, which can result in the wrong field being accessed by a getter or setter.

This rule raises an issue in any of these cases:

- A setter does not update the field with the corresponding name.
- A getter does not access the field with the corresponding name.

Noncompliant Code Example

```
class A {
  private int x;
  private int y;

  public void setX(int val) { // Noncompliant: field 'x' is
    this.y = val;
  }

  public int getY() { // Noncompliant: field 'y' is not used
    return this.x;
  }
}
```

Compliant Solution

```
class A {
  private int x;
  private int y;

  public void setX(int val) {
    this.x = val;
  }

  public int getY() {
    return this.y;
  }
}
```

Available In:

sonarlint | sonarcloud | sonarqube

OpenSAML2 should be configured to prevent authentication bypass

Vulnerability

Server-side requests should not be vulnerable to forging attacks

Vulnerability

Collections should not be modified while they are iterated

Bug

Equals method should be overridden in records containing array fields

Bug