sonar RULES

Products

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...

bits-1
Bug

"compareTo" should not return "Integer.MIN_VALUE"
Bug

Boxing and unboxing should not be immediately reversed
Bug

"equals(Object obj)" should test argument type
Bug

"Serializable" inner classes of non-serializable classes should be "static"
Bug

The non-serializable super class of a "Serializable" class should have a no-argument constructor
Bug

Method parameters, caught exceptions and foreach variables' initial values should not be ignored
Bug

"equals(Object obj)" and "hashCode()" should be overridden in pairs
Bug

Disclosing fingerprints from web application technologies is security-sensitive
Security Hotspot

Having a permissive Cross-Origin Resource Sharing policy is security-sensitive
Security Hotspot

Delivering code in production with debug features activated is security-sensitive
Security Hotspot

Child class methods named for parent class methods should be overrides

Analyze your code

Bug Major pitfall

When a method in a child class has the same signature as a method in a parent class, it is assumed to be an override. However, that's not the case when:

- the parent class method is `static` and the child class method is not.
- the arguments or return types of the child method are in different packages than those of the parent method.
- the parent class method is `private`.

Typically, these things are done unintentionally; the private parent class method is overlooked, the `static` keyword in the parent declaration is overlooked, or the wrong class is imported in the child. But if the intent is truly for the child class method to be different, then the method should be renamed to prevent confusion.

Noncompliant Code Example

```
// Parent.java
import computer.Pear;
public class Parent {

  public void doSomething(Pear p) {
    //,,,
  }

  public static void doSomethingElse() {
    //...
  }
}

// Child.java
import fruit.Pear;
public class Child extends Parent {

  public void doSomething(Pear p) {  // Noncompliant; this i
    // ...
  }


  public void doSomethingElse() {  // Noncompliant; parent m
    //...
  }
}
```

Compliant Solution

```
// Parent.java
import computer.Pear;
public class Parent {

  public void doSomething(Pear p) {
```

Searching OS commands in PATH is security-sensitive

Security Hotspot

Allowing both safe and unsafe HTTP methods is security-sensitive

Security Hotspot

Creating cookies without the "HttpOnly" flag is security-sensitive

Security Hotspot

Creating cookies without the "secure" flag is security-sensitive

Security Hotspot

```
    //,,,
  }

  public static void doSomethingElse() {
    //...
  }
}

// Child.java
import computer.Pear;  // import corrected
public class Child extends Parent {

  public void doSomething(Pear p) {  // true override (see i
    //,,,
  }

  public static void doSomethingElse() {
    //...
  }
}
```

Available In:

sonarlint sonarcloud sonarqube