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...

Enumeration should not be implemented
Code Smell

Synchronized classes Vector, Hashtable, Stack and StringBuffer should not be used
Code Smell

Unused "private" methods should be removed
Code Smell

Try-catch blocks should not be nested
Code Smell

Track uses of "FIXME" tags
Code Smell

Deprecated elements should have both the annotation and the Javadoc tag
Code Smell

Assignments should not be made from within sub-expressions
Code Smell

Generic exceptions should never be thrown
Code Smell

Labels should not be used
Code Smell

Utility classes should not have public constructors
Code Smell

Local variables should not shadow class fields
Code Smell

Redundant pairs of parentheses should be removed
Code Smell

Raw byte values should not be used in bitwise operations in combination with shifts

Analyze your code

Bug Major cert

When reading bytes in order to build other primitive values such as `int`s or `long`s, the `byte` values are automatically promoted, but that promotion can have unexpected results.

For instance, the binary representation of the integer 640 is `0b0000_0010_1000_0000`, which can also be written with the array of (unsigned) bytes `[2, 128]`. However, since Java uses two's complement, the representation of the integer in signed bytes will be `[2, -128]` (because the `byte` `0b1000_0000` is promoted to the `int` `0b1111_1111_1111_1111_1111_1111_1000_0000`). Consequently, trying to reconstruct the initial integer by shifting and adding the values of the bytes without taking care of the sign will not produce the expected result.

To prevent such accidental value conversion, use bitwise and (`&`) to combine the `byte` value with `0xff` (255) and turn all the higher bits back off.

This rule raises an issue any time a `byte` value is used as an operand without `& 0xff`, when combined with shifts.

Noncompliant Code Example

```
int intFromBuffer() {
  int result = 0;
  for (int i = 0; i < 4; i++) {
    result = (result << 8) | readByte(); // Noncompliant
  }
  return result;
}
```

Compliant Solution

```
int intFromBuffer() {
  int result = 0;
  for (int i = 0; i < 4; i++) {
    result = (result << 8) | (readByte() & 0xff);
  }
  return result;
}
```

See

- CERT, NUM52-J. - Be aware of numeric promotion behavior

Available In:

sonarlint sonarcloud sonarqube

Inheritance tree of classes should not be too deep

Code Smell

Nested blocks of code should not be left empty

Code Smell

Methods should not have too many parameters

Code Smell

Unused "private" fields should be removed