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...

"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

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

Using hardcoded IP addresses is security-sensitive
Security Hotspot

'serialVersionUID' field should not be set to '0L' in records
Code Smell

Permitted types of a sealed class should be omitted if they are declared in the same file
Code Smell

"InterruptedException" should not be ignored

Analyze your code

Bug Major cwe error-handling multi-threading

`InterruptedExceptions` should never be ignored in the code, and simply logging the exception counts in this case as "ignoring". The throwing of the `InterruptedException` clears the interrupted state of the Thread, so if the exception is not handled properly the information that the thread was interrupted will be lost. Instead, `InterruptedExceptions` should either be rethrown - immediately or after cleaning up the method's state - or the thread should be re-interrupted by calling `Thread.interrupt()` even if this is supposed to be a single-threaded application. Any other course of action risks delaying thread shutdown and loses the information that the thread was interrupted - probably without finishing its task.

Similarly, the `ThreadDeath` exception should also be propagated. According to its JavaDoc:

> If `ThreadDeath` is caught by a method, it is important that it be rethrown so that the thread actually dies.

Noncompliant Code Example

```
public void run () {
  try {
    while (true) {
      // do stuff
    }
  }catch (InterruptedException e) { // Noncompliant; logging
    LOGGER.log(Level.WARN, "Interrupted!", e);
  }
}
```

Compliant Solution

```
public void run () {
  try {
    while (true) {
      // do stuff
    }
  }catch (InterruptedException e) {
    LOGGER.log(Level.WARN, "Interrupted!", e);
    // Restore interrupted state...
    Thread.currentThread().interrupt();
  }
}
```

See

- MITRE, CWE-391 - Unchecked Error Condition

Available In:

Switch arrow labels should not use redundant keywords

Code Smell

Text blocks should not be used in complex expressions

Code Smell

Pattern Matching for "instanceof" operator should be used instead of simple "instanceof" + cast

Code Smell

Call to Mockito method "verify", "when" or "given" should be simplified

sonarlint sonarcloud sonarqube