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...

Basic authentication should not be used
Vulnerability

Regular expressions should not be vulnerable to Denial of Service attacks
Vulnerability

"HttpServletRequest.getRequestedSessio should not be used
Vulnerability

Hashes should include an unpredictable salt
Vulnerability

Calls to methods should not trigger an IllegalArgumentException
Bug

Unsupported methods should not be called on some collection implementations
Bug

Cast operations should not trigger a ClassCastException
Bug

Members ignored during record serialization should not be used
Bug

Map "computeIfAbsent()" and "computeIfPresent()" should not be used to add "null" values.
Bug

Regex lookahead assertions should not be contradictory
Bug

Back references in regular expressions should only refer to capturing groups that are matched before the reference

Assertions should be complete

Analyze your code

Code Smell Blocker tests assertj mockito

It is very easy to write incomplete assertions when using some test frameworks. This rule enforces complete assertions in the following cases:

- Fest: `assertThat` is not followed by an assertion invocation
- AssertJ: `assertThat` is not followed by an assertion invocation
- Mockito: `verify` is not followed by a method invocation
- Truth: `assertXXX` is not followed by an assertion invocation

In such cases, what is intended to be a test doesn't actually verify anything

Noncompliant Code Example

```
// Fest
boolean result = performAction();
// let's now check that result value is true
assertThat(result); // Noncompliant; nothing is actually che

// Mockito
List mockedList = Mockito.mock(List.class);
mockedList.add("one");
mockedList.clear();
// let's check that "add" and "clear" methods are actually c
Mockito.verify(mockedList); // Noncompliant; nothing is chec
```

Compliant Solution

```
// Fest
boolean result = performAction();
// let's now check that result value is true
assertThat(result).isTrue();

// Mockito
List mockedList = Mockito.mock(List.class);
mockedList.add("one");
mockedList.clear();
// let's check that "add" and "clear" methods are actually c
Mockito.verify(mockedList).add("one");
Mockito.verify(mockedList).clear();
```

Exceptions

Variable assignments and return statements are skipped to allow helper methods.

```
private BooleanAssert check(String filename, String key) {
  String fileContent = readFileContent(filename);
  performReplacements(fileContent);
  return assertThat(fileContent.contains(key)); // No issue
}
```

Bug

Regex boundaries should not be used in a way that can never be matched

Bug

Regex patterns following a possessive quantifier should not always fail

Bug

Regular expressions should be syntactically valid

Bug

Assertions comparing incompatible

```
@Test
public void test() {
  check("foo.txt", "key1").isTrue();
  check("bar.txt", "key2").isTrue();
}
```

Available In:

sonarlint | sonarcloud | sonarqube