Java static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your JAVA code

Vulnerability

Reflection should not be vulnerable to injection attacks
Vulnerability

Authorizations should be based on strong decisions
Vulnerability

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

Reflection should not be used to increase accessibility of records' fields
Bug

AssertJ assertions with "Consumer" arguments should contain assertion inside consumers
Bug

The regex escape sequence \cX should only be used with characters in the @-_ range
Bug

Regular expressions should not overflow the stack
Bug

Tests method should not be annotated with competing

"Random" objects should be reused

Analyze your code

Bug Critical owasp

Creating a new `Random` object each time a random value is needed is inefficient and may produce numbers which are not random depending on the JDK. For better efficiency and randomness, create a single `Random`, then store, and reuse it.

The `Random()` constructor tries to set the seed with a distinct value every time. However there is no guarantee that the seed will be random or even uniformly distributed. Some JDK will use the current time as seed, which makes the generated numbers not random at all.

This rule finds cases where a new `Random` is created each time a method is invoked.

Noncompliant Code Example

```
public void doSomethingCommon() {
  Random rand = new Random();  // Noncompliant; new instance
  int rValue = rand.nextInt();
  //...
```

Compliant Solution

```
private Random rand = SecureRandom.getInstanceStrong();  //

public void doSomethingCommon() {
  int rValue = this.rand.nextInt();
  //...
```

Exceptions

A class which uses a `Random` in its constructor or in a static `main` function and nowhere else will be ignored by this rule.

See

- OWASP Top 10 2017 Category A6 - Security Misconfiguration

Available In:
sonarlint sonarcloud sonarqube

annotations

Bug

Assertions should not be used in production code

Bug

DateTimeFormatters should not use mismatched year and week numbers

Bug

Unicode Grapheme Clusters should be avoided inside regex character classes

Bug