Java static code analysis

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

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

Switch arrow labels should not use redundant keywords
Code Smell

Classes extending java.lang.Thread should override the "run" method

Analyze your code

Bug Major multi-threading pitfall

According to the Java API documentation:

> There are two ways to create a new thread of execution. One is to declare a class to be a subclass of Thread. This subclass should override the run method of class Thread. An instance of the subclass can then be allocated and started...
> The other way to create a thread is to declare a class that implements the Runnable interface. That class then implements the run method. An instance of the class can then be allocated, passed as an argument when creating Thread, and started.

By definition, extending the Thread class without overriding the `run` method doesn't make sense, and implies that the contract of the `Thread` class is not well understood.

Noncompliant Code Example

```
public class MyRunner extends Thread { // Noncompliant; run

  public void doSometing() {...}
}
```

Exceptions

If `run()` is not overridden in a class extending `Thread`, it means that starting the thread will actually call `Thread.run()`. However, `Thread.run()` does nothing if it has not been fed with a target `Runnable`. The rule consequently ignore classes extending `Thread` if they are calling, in their constructors, the `super(...)` constructor with a proper `Runnable` target.

```
class MyThread extends Thread { // Compliant - calling super
  MyThread(Runnable target) {
    super(target); // calling super constructor with a Runna
    // ...
  }
}
```

Available In:

sonarlint sonarcloud sonarqube

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

Code Smell

Character classes should be preferred over reluctant quantifiers in regular expressions