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...

Abstract class names should comply with a naming convention
Code Smell

Strings literals should be placed on the left side when checking for equality
Code Smell

Files should contain an empty newline at the end
Code Smell

Source code should be indented consistently
Code Smell

A close curly brace should be located at the beginning of a line
Code Smell

Close curly brace and the next "else", "catch" and "finally" keywords should be on two different lines
Code Smell

Close curly brace and the next "else", "catch" and "finally" keywords should be located on the same line
Code Smell

An open curly brace should be located at the beginning of a line
Code Smell

An open curly brace should be located at the end of a line
Code Smell

Tabulation characters should not be used
Code Smell

Functions should not be defined with a variable number of arguments
Code Smell

Searching OS commands in PATH is security-sensitive

Analyze your code

Security Hotspot   Minor   cwe owasp

When executing an OS command and unless you specify the full path to the executable, then the locations in your application's `PATH` environment variable will be searched for the executable. That search could leave an opening for an attacker if one of the elements in `PATH` is a directory under his control.

Ask Yourself Whether

- The directories in the PATH environment variable may be defined by not trusted entities.

There is a risk if you answered yes to this question.

Recommended Secure Coding Practices

Fully qualified/absolute path should be used to specify the OS command to execute.

Sensitive Code Example

The full path of the command is not specified and thus the executable will be searched in all directories listed in the `PATH` environment variable:

```
Runtime.getRuntime().exec("make");  // Sensitive
Runtime.getRuntime().exec(new String[]{"make"});  // Sensiti

ProcessBuilder builder = new ProcessBuilder("make");  // Sen
builder.command("make");  // Sensitive
```

Compliant Solution

The command is defined by its full path:

```
Runtime.getRuntime().exec("/usr/bin/make");  // Compliant
Runtime.getRuntime().exec(new String[]{"~/bin/make"});  // C

ProcessBuilder builder = new ProcessBuilder("./bin/make");
builder.command("../bin/make");  // Compliant
builder.command(Arrays.asList("..\bin\make", "-j8")); // Com

builder = new ProcessBuilder(Arrays.asList(".\make"));  // C
builder.command(Arrays.asList("C:\bin\make", "-j8"));  // Co
builder.command(Arrays.asList("\\SERVER\bin\make"));  // Com
```

See

- OWASP Top 10 2021 Category A8 - Software and Data Integrity Failures
- OWASP Top 10 2017 Category A1 - Injection
- MITRE, CWE-426 - Untrusted Search Path
- MITRE, CWE-427 - Uncontrolled Search Path Element

Local-Variable Type Inference should be used

Code Smell

Migrate your tests from JUnit4 to the new JUnit5 annotations

Code Smell

Track uses of disallowed classes

Code Smell

Track uses of "@SuppressWarnings" annotations

Code Smell

Available In:

sonarcloud | sonarqube