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

Creating cookies without the "HttpOnly" flag is security-sensitive

Analyze your code

Security Hotspot   Minor   cwe sans-top25 privacy owasp

When a cookie is configured with the `HttpOnly` attribute set to *true*, the browser guaranties that no client-side script will be able to read it. In most cases, when a cookie is created, the default value of `HttpOnly` is *false* and it's up to the developer to decide whether or not the content of the cookie can be read by the client-side script. As a majority of Cross-Site Scripting (XSS) attacks target the theft of session-cookies, the `HttpOnly` attribute can help to reduce their impact as it won't be possible to exploit the XSS vulnerability to steal session-cookies.

Ask Yourself Whether

- the cookie is sensitive, used to authenticate the user, for instance a *session-cookie*
- the `HttpOnly` attribute offer an additional protection (not the case for an *XSRF-TOKEN cookie* / CSRF token for example)

There is a risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

- By default the `HttpOnly` flag should be set to *true* for most of the cookies and it's mandatory for session / sensitive-security cookies.

Sensitive Code Example

If you create a security-sensitive cookie in your JAVA code:

```
Cookie c = new Cookie(COOKIENAME, sensitivedata);
c.setHttpOnly(false);  // Sensitive: this sensitive cookie i
```

By default the `HttpOnly` flag is set to *false:*

```
Cookie c = new Cookie(COOKIENAME, sensitivedata);  // Sensit
```

Compliant Solution

```
Cookie c = new Cookie(COOKIENAME, sensitivedata);
c.setHttpOnly(true); // Compliant: this sensitive cookie is
```

See

- OWASP Top 10 2021 Category A5 - Security Misconfiguration
- OWASP HttpOnly
- OWASP Top 10 2017 Category A7 - Cross-Site Scripting (XSS)
- MITRE, CWE-1004 - Sensitive Cookie Without 'HttpOnly' Flag
- SANS Top 25 - Insecure Interaction Between Components
- Derived from FindSecBugs rule HTTPONLY_COOKIE

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