




 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 rules632

Vulnerability53

Bug154

Security Hotspot36

Code Smell389

Quick Fix42

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

Delivering code in production with debug features activated is security-sensitive

Analyze your code

Security Hotspot

Minor ?

cwe error-handling spring debug user-experience owasp

Delivering code in production with debug features activated is security-sensitive. It has led in the past to the following vulnerabilities:

- [CVE-2018-1999007](#)
- [CVE-2015-5306](#)
- [CVE-2013-2006](#)

An application's debug features enable developers to find bugs more easily and thus facilitate also the work of attackers. It often gives access to detailed information on both the system running the application and users.

Ask Yourself Whether

- the code or configuration enabling the application debug features is deployed on production servers or distributed to end users.
- the application runs by default with debug features activated.

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

Recommended Secure Coding Practices

Do not enable debug features on production servers or applications distributed to end users.

Sensitive Code Example

Throwable.printStackTrace(...) prints a Throwable and its stack trace to System.Err (by default) which is not easily parseable and can expose sensitive information:

```
try {
    /* ... */
} catch(Exception e) {
    e.printStackTrace(); // Sensitive
}
```





[EnableWebSecurity](#) annotation for SpringFramework with debug to true enables debugging support:

```
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.co

@Configuration
@EnableWebSecurity(debug = true) // Sensitive
public class WebSecurityConfig extends WebSecurityConfigurer
// ...
}
```

https://rules.sonarsource.com/java/RSPEC-4507

1/2

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

[WebView.setWebContentsDebuggingEnabled\(true\)](#) for Android enables debugging support:

```
import android.webkit.WebView;

WebView.setWebContentsDebuggingEnabled(true); // Sensitive
WebView.getFactory().getStatics().setWebContentsDebuggingEna
```

Compliant Solution

Loggers should be used (instead of printStackTrace) to print throwables:

```
try {
    /* ... */
} catch (Exception e) {
    LOGGER.log("context", e);
}
```

[EnableWebSecurity](#) annotation for SpringFramework with debug to false disables debugging support:

```
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.co

@Configuration
@EnableWebSecurity(debug = false)
public class WebSecurityConfig extends WebSecurityConfigurer
```

[WebView.setWebContentsDebuggingEnabled\(false\)](#) for Android disables debugging support:

```
import android.webkit.WebView;

WebView.setWebContentsDebuggingEnabled(false);
WebView.getFactory().getStatics().setWebContentsDebuggingEna
```

See

- [OWASP Top 10 2021 Category A5](#) - Security Misconfiguration
- [OWASP Top 10 2017 Category A3](#) - Sensitive Data Exposure
- [MITRE, CWE-489](#) - Active Debug Code
- [MITRE, CWE-215](#) - Information Exposure Through Debug Information

Available In:  