

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...

Code Smell

"catch" clauses should do more than rethrow

Mutable fields should not be "public static"

The diamond operator ("<>") should be used

"finalize" should not set fields to "null"

Subclasses that add fields should override "equals"

Catches should be combined

Methods of "Random" that return floating point values should not be used in random integer generation

Parsing should be used to convert "Strings" to primitives

Classes should not be empty

Fields in non-serializable classes should not be "transient"

Boolean checks should not be inverted

Redundant casts should not be used

Enabling JavaScript support for WebViews is security-sensitive

Analyze your code

Security Hotspot

Major

cwe owasp android

WebViews can be used to display web content as part of a mobile application. A browser engine is used to render and display the content. Like a web application a mobile application that uses WebViews can be vulnerable to Cross-Site Scripting if untrusted code is rendered. In the context of a WebView JavaScript code can exfiltrate local files that might be sensitive or even worse, access exposed functions of the application that can result in more severe vulnerabilities such as code injection. Thus JavaScript support should not be enabled for WebViews unless it is absolutely necessary and the authenticity of the web resources can be guaranteed.

Ask Yourself Whether

- The WebView only renders static web content that does not require JavaScript code to be executed.
- The WebView contains untrusted data that could cause harm when rendered.

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

Recommended Secure Coding Practices

It's recommended to disable JavaScript support for WebViews unless it is necessary to execute JavaScript code. Only trusted pages should be rendered.

Sensitive Code Example

```
import android.webkit.WebView;

WebView webView = (WebView) findViewById(R.id.webview);
webView.getSettings().setJavaScriptEnabled(true); // Sensitive
```

Compliant Solution

```
import android.webkit.WebView;

WebView webView = (WebView) findViewById(R.id.webview);
webView.getSettings().setJavaScriptEnabled(false);
```

See





- OWASP Top 10 2021 Category A3 - Injection
- OWASP Top 10 2017 Category A6 - Security Misconfiguration
- OWASP Top 10 2017 Category A7 - Cross-Site Scripting (XSS)
- MITRE, CWE-79 - Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

Available In:

sonarcloudsonarqube

https://rules.sonarsource.com/java/RSPEC-6362

1/2

<div>redundant casts should not be used</div> <div> Code Smell</div>	
<div>"@Deprecated" code should not be used</div> <div> Code Smell</div>	
<div>"toString()" should never be called on a String object</div> <div> Code Smell</div>	
<div>Annotation repetitions should not be wrapped</div> <div> Code Smell</div>	
<div>Multiple variables should not be</div>	

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.

[Privacy Policy](#)