Secrets
ABAP
Apex
C
C++
CloudFormation
COBOL
C#
CSS
Flex
Go
HTML
Java
JavaScript
Kotlin
Kubernetes
Objective C
PHP
PL/I
PL/SQL
Python
RPG
Ruby
Scala
Swift
Terraform
Text
TypeScript
T-SQL
VB.NET
VB6
XML

XML static code analysis

Unique rules to find Bugs and Code Smells in your XML code

All rules 36 | Vulnerability 6 | Bug 5 | Security Hotspot 9 | Code Smell 16

Tags

Search by name...

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

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

Deprecated "${pom}" properties should not be used
Code Smell

Track uses of "TODO" tags
Code Smell

EJB interceptor exclusions should be declared as annotations
Code Smell

Track uses of disallowed dependencies
Code Smell

Newlines should follow each element
Code Smell

XML parser failure
Code Smell

Track breaches of an XPath rule
Code Smell

Lines should not be too long
Code Smell

pom elements should be in the recommended order
Code Smell

Artifact ids should follow a naming convention
Code Smell

Group ids should follow a naming convention
Code Smell

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

Analyze your code

Security Hotspot | Minor | cwe error-handling debug android user-experience owasp

In the application manifest element of an android application, setting debuggable property to `true` could introduce a security risk.

It's more easy to perform reverse engineering and inject arbitrary code in the context of a debuggable application.

Ask Yourself Whether

- the development of the app is completed and the `debuggable` property is set to *true*
- the app will be published on the Play Store or distributed in any other ways and the `debuggable` property is set to *true*

You are at risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

It is not recommended to release debuggable application. Avoid hardcoding the debug mode in the manifest because the build tool will add the property automatically and assign the correct value depending on the build type.

Sensitive Code Example

In `AndroidManifest.xml` the android debuggable property is set to `true`:

```
<application
  android:icon="@mipmap/ic_launcher"
  android:label="@string/app_name"
  android:roundIcon="@mipmap/ic_launcher_round"
  android:supportsRtl="true"
  android:debuggable="true"
  android:theme="@style/AppTheme">
</application>  <!-- Sensitive -->
```

Compliant Solution

In `AndroidManifest.xml` the android debuggable property is set to `false`:

```
<application
  android:icon="@mipmap/ic_launcher"
  android:label="@string/app_name"
  android:roundIcon="@mipmap/ic_launcher_round"
  android:supportsRtl="true"
  android:debuggable="false"
  android:theme="@style/AppTheme">
</application> <!-- Compliant -->
```

See

- OWASP Top 10 2021 Category A5 - Security Misconfiguration
- Mobile AppSec Verification Standard - Code Quality and Build Setting Requirements
- OWASP Mobile Top 10 2016 Category M10 - Extraneous Functionality
- MITRE, CWE-215 - Information Exposure Through Debug Information
- developer.android.com - Prepare for release

Available In:

sonarcloud | sonarqube