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

Kotlin static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your KOTLIN code

All rules 98 | Vulnerability 10 | Bug 17 | Security Hotspot 15 | Code Smell 56

Tags

Search by name...

Hard-coded credentials are security-sensitive
Security Hotspot

Cipher algorithms should be robust
Vulnerability

Encryption algorithms should be used with secure mode and padding scheme
Vulnerability

Server hostnames should be verified during SSL/TLS connections
Vulnerability

Server certificates should be verified during SSL/TLS connections
Vulnerability

Cryptographic keys should be robust
Vulnerability

Weak SSL/TLS protocols should not be used
Vulnerability

"SecureRandom" seeds should not be predictable
Vulnerability

Cipher Block Chaining IVs should be unpredictable
Vulnerability

Hashes should include an unpredictable salt
Vulnerability

Regular expressions should be syntactically valid
Bug

"runFinalizersOnExit" should not be called
Bug

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 WebWiew 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

val webView: WebView = findViewById(R.id.webview)
webView.getSettings().setJavaScriptEnabled(true) // Sens
```

Compliant Solution

```
import android.webkit.WebView

val 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:

sonarcloud | sonarqube

"ScheduledThreadPoolExecutor" should not have 0 core threads

Bug

Jump statements should not occur in "finally" blocks

Bug

Using clear-text protocols is security-sensitive

Security Hotspot

Accessing Android external storage is security-sensitive

Security Hotspot

Receiving intents is security-sensitive

Security Hotspot

Broadcasting intents is security-sensitive

Security Hotspot

Using weak hashing algorithms is security-sensitive

Security Hotspot

Using pseudorandom number generators (PRNGs) is security-sensitive

Security Hotspot

Empty lines should not be tested with regex MULTILINE flag

Code Smell

Cognitive Complexity of functions should not be too high

Code Smell