Code Smell (56)





Swift

Terraform

Scala

Text

TypeScrir

TypeScript
T-SQL

VB VB.NET

VB6 VB6

XML XML



All rules (98)

## **Kotlin static code analysis**

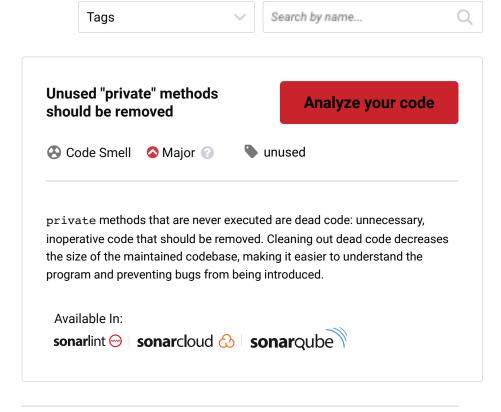
6 Vulnerability (10)

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

**♣** Bug 17

Hard-coded credentials are security-
sensitive
Security Hotspot
Cipher algorithms should be robust
<b>♠</b> Vulnerability
Encryption algorithms should be used with secure mode and padding scheme
<b>읍</b> Vulnerability
Server hostnames should be verified during SSL/TLS connections
Carability
Server certificates should be verified during SSL/TLS connections
❸ Vulnerability
Cryptographic keys should be robust
Weak SSL/TLS protocols should not be used
❸ Vulnerability
"SecureRandom" seeds should not be predictable
Cipher Block Chaining IVs should be unpredictable
Hashes should include an unpredictable salt
Carability
Regular expressions should be syntactically valid
<b>∰</b> Bug
"runFinalizersOnExit" should not be called

Rug Bug



Security Hotspot (15)

© 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

"ScheduledThreadPoolExecutor" should not have 0 core threads
Jump statements should not occur in "finally" blocks
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