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

Related "if/else if" statements should not have the same condition

Analyze your code

Bug | Major | unused pitfall

A chain of `if/else if` statements is evaluated from top to bottom. At most, only one branch will be executed: the first one with a condition that evaluates to `true`.

Therefore, duplicating a condition automatically leads to dead code. Usually, this is due to a copy/paste error. At best, it's simply dead code and at worst, it's a bug that is likely to induce further bugs as the code is maintained, and obviously it could lead to unexpected behavior.

Available In:

sonarlint | 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