



# **Kotlin static code analysis**

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

**₩** Bug 17

Tags

All rules 98	rability 10
Hard-coded credentials are sensitive	security-
Security Hotspot	
Cipher algorithms should be	e robust
★ Vulnerability	
Encryption algorithms shou with secure mode and padd scheme	
C Vulnerability	
Server hostnames should be during SSL/TLS connections	
★ Vulnerability	
Server certificates should be during SSL/TLS connections	
<b>☆</b> Vulnerability	
Cryptographic keys should be Vulnerability	oe robust
Weak SSL/TLS protocols sh	ould not
<b>⋒</b> Vulnerability	
"SecureRandom" seeds sho predictable	uld not be
<b>☆</b> Vulnerability	
Cipher Block Chaining IVs s unpredictable	hould be
<b>G</b> Vulnerability	
Hashes should include an unpredictable salt	
<b>6</b> Vulnerability	
Regular expressions should syntactically valid	be
Rug	
"runFinalizersOnExit" should called	l not be
👚 Bug	

~	Search by name	Q

Code Smell 56

Security Hotspot (15)

"ScheduledThreadPoolExecutor" should not have 0 core threads

📆 Bug

Jump statements should not occur in "finally" blocks

📆 Bug

Using clear-text protocols is securitysensitive

Security Hotspot

Accessing Android external storage is security-sensitive

Security Hotspot

Receiving intents is security-sensitive

Security Hotspot

Broadcasting intents is securitysensitive

Security Hotspot

Using weak hashing algorithms is security-sensitive

Security Hotspot

Using pseudorandom number generators (PRNGs) is securitysensitive

Security Hotspot

Empty lines should not be tested with regex MULTILINE flag

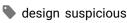
Code Smell

**Cognitive Complexity of functions** should not be too high

Code Smell

## Two branches in a conditional structure should not have exactly the same implementation

## Analyze your code



Having two clauses in a when statement or two branches in an if chain with the same implementation is at best duplicate code, and at worst a coding error. If the same logic is truly needed for both instances, then in an if chain they should be combined, or for a when, duplicates should be refactored.

### **Noncompliant Code Example**

```
fun s1871(x: Int) {
   when (x) {
       1 -> {
            val y = x / 2
            print(y)
        }
        2 -> {
            val y = x / 2
           print(y)
        }
   }
}
```

#### **Exceptions**

Blocks in an if chain that contain a single line of code are ignored, as are blocks in a when statement that contain a single line of code with or without a following break.

```
if (a == 1) {
   doSomething() //no issue, usually this is done on p
} else if (a == 2) {
    doSomethingElse()
} else {
    doSomething()
}
```

But this exception does not apply to  ${\tt if}$  chains without else-s, or to when-es without else clauses when all branches have the same single line of code. In case of if chains with else-s, or of when-es with default clauses, rule {rule:kotlin:S3923} raises a bug.

```
if (a == 1) {
 doSomething() //Noncompliant, this might have been do
} else if (a == 2) {
  doSomething()
```

Available In:

sonarlint ⊕ | sonarcloud ♦ | sonarqube

© 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