

































-  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

Expressions should not be too complex

Analyze your code

 Code Smell    Critical       brain-overload

The complexity of an expression is defined by the number of &&, || and condition ? ifTrue : ifFalse operators it contains.

A single expression's complexity should not become too high to keep the code readable.

### Noncompliant Code Example

With the default threshold value of 3:

```
if (((condition1 && condition2) || (condition3 && condit
```

### Compliant Solution

```
if ((myFirstCondition() || mySecondCondition()) && myLas
```

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](#)

<div>"ScheduledThreadPoolExecutor" should not have 0 core threads</div> <div> Bug</div>
<div>Jump statements should not occur in "finally" blocks</div> <div> Bug</div>
<div>Using clear-text protocols is security-sensitive</div> <div> Security Hotspot</div>
<div>Accessing Android external storage is security-sensitive</div> <div> Security Hotspot</div>
<div>Receiving intents is security-sensitive</div> <div> Security Hotspot</div>
<div>Broadcasting intents is security-sensitive</div> <div> Security Hotspot</div>
<div>Using weak hashing algorithms is security-sensitive</div> <div> Security Hotspot</div>
<div>Using pseudorandom number generators (PRNGs) is security-sensitive</div> <div> Security Hotspot</div>
<div>Empty lines should not be tested with regex MULTILINE flag</div> <div> Code Smell</div>
<div>Cognitive Complexity of functions should not be too high</div> <div> Code Smell</div>