

-  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

Functions should not have too many parameters

Analyze your code

 Code Smell  Major   brain-overload

A long parameter list can indicate that a new structure should be created to wrap the numerous parameters or that the function is doing too many things.

Noncompliant Code Example

With a maximum number of 4 parameters:

```
fun foo(p1: String, p2: String, p3: String, p4: String,
        // ...
    )
{
}
```

Compliant Solution

```
fun foo(p1: String, p2: String, p3: String, p4: String)
    // ...
{
}
```

Exceptions

Methods annotated with Spring's `@RequestMapping` (and related shortcut annotations, like `@GetRequest`) or `@JsonCreator` may have a lot of parameters, encapsulation being possible. Such methods are therefore ignored.

Available In:

sonarlint  | **sonarcloud**  | **sonarqube** 

<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>