Kotlin static code analysis

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

| 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 |

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

ViewModel classes should create coroutines

Analyze your code

☢ Code Smell   🔻 Major ?   🏷 coroutines android bad-practice

Views should not be responsible for directly triggering coroutines. Hence, `ViewModel` classes should prefer creating coroutines instead of exposing suspending functions to perform some piece of business logic. This allows for easier testing of your application, as `ViewModel` classes can be unit tested, whereas views require instrumentation tests.

Please refer to the Android docs for more advanced examples and mechanisms of updating the views with data generated asynchronously.

This rule raises an issue when suspending functions are exposed by classes inheriting from `ViewModel`.

Noncompliant Code Example

```
class MyViewModel : ViewModel() {
    suspend fun performAction() = suspendingWorker()
}
```

Compliant Solution

```
class MyViewModel : ViewModel() {
    fun performAction() =
        viewModelScope.launch {
            suspendingWorker()
        }
}
```

See

- The ViewModel should create coroutines (Android coroutines best practices)

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