sonar RULES Products

- 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

Flow intermediate operation results should not be left unused

Analyze your code

Bug | Major | coroutines

In Kotlin, `Flow` represents a cold stream concept. Similar to `Stream` in Java or `Sequence` in Kotlin, we can manipulate the data inside the flow (filter, transform, collect, etc). The `Flow` API, just like `Stream` and `Sequence`, offers two types of operations: intermediate and terminal. Intermediate operations again return a `Flow` instance, all other operations are considered terminal. As flows are naturally lazy, no operations will actually be started until a terminal operation is called.

This rule reports an issue when the result of an intermediate operation on `Flow` is left unused.

Noncompliant Code Example

```
suspend fun main() {
    val flow = flow {
        emit(1)
        emit(2)
        emit(3)
    }

    flow.take(2) // Noncompliant, the result of this ope
}
```

Compliant Solution

```
suspend fun main() {
    val flow = flow {
        emit(1)
        emit(2)
        emit(3)
    }

    flow.take(2).collect { println(it) } // Compliant, c
}
```

See

- Flow documentation

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