Kotlin static code analysis

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

Kotlin coroutines API for timeouts should be used

Analyze your code

Code Smell Major coroutines

Sometimes there is the need to cancel the execution of a coroutine after a given period of time. You can do this manually by combining the `delay()` and `cancel()` functions. However, this technique is verbose and error-prone. An easier way to manage timeouts is using the function `withTimeout()` or `withTimeoutOrNull()`.

The `withTimeout` function will throw a `TimeoutCancellationException` when the timeout is reached, while `withTimeoutOrNull` will simply return `null` instead.

This rule raises an issue if timeout mechanisms are implemented manually instead of using appropriate built-in functions.

Noncompliant Code Example

```
suspend fun main() {
    coroutineScope {
        val job = launch {
            delay(2000L)
            println("Finished")
        }
        delay(500L)
        job.cancel()
    }
}
```

Compliant Solution

```
suspend fun main() {
    coroutineScope {
        withTimeoutOrNull(1000L){
            delay(2000L)
            println("Finished")
        }
    }
}
```

See

- Cancellation and timeouts

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