

- 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

"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

Native features should be preferred to Guava

Analyze your code

Code Smell Major guava

Some Guava features were really useful for Java 7 application because Guava was bringing APIs missing in the JDK. The introduction of Kotlin and Java 8+ fixed some of these limitations. When migrating an application to Kotlin and/or using a Java 8+ target, it is recommended to prefer the provided native APIs over Guava to ease maintenance: developers don't need to learn how to use two APIs and can stick to the default one.

This rule raises an issue when the Guava APIs listed below are used.









| Guava API                                  | Kotlin API   |
|--|--|
| com.google.common.base.Joiner.on           | kotlin.collections.Iterable.joinToString or kotlin.Array.joinToString  |
| com.google.common.base.Predicate           | Use function with type (T) → Boolean as a replacement for Predicate<T> |
| com.google.common.base.Function            | Use function with type (T) → R as a replacement for Function<T, R>     |
| com.google.common.base.Supplier            | Use function with type () → T as a replacement for Supplier<T>         |
| com.google.common.io.Files.createTempDir   | kotlin.io.path.createTempDirectory                                     |
| com.google.common.collect.ImmutableSet.of  | kotlin.collections.setOf   |
| com.google.common.collect.ImmutableList.of | kotlin.collections listOf  |
| com.google.common.collect.ImmutableMap.of  | kotlin.collections.mapOf   |

| Guava API                                    | Java 8 API                     |
|--|--------------------------------|
| com.google.common.io.BaseEncoding.base64     | java.util.Base64               |
| com.google.common.io.BaseEncoding.base64Url  | java.util.Base64               |
| com.google.common.base.Optional              | java.util.Optional*            |
| com.google.common.base.Optional.of           | java.util.Optional.of*         |
| com.google.common.base.Optional.absent       | java.util.Optional.empty*      |
| com.google.common.base.Optional.fromNullable | java.util.Optional.ofNullable* |

\*: Note that this rule will also raise issues for the use of Guava Optional and recommend using the Java 8+ equivalent instead. In most Kotlin-only cases, you will probably be better off using Kotlin's null-safe type system directly, without wrapping Optional constructs. In some cases, such as those involving Java interoperability, it may be necessary to use Optional in Kotlin, however.

Available In:

sonarlint | sonarcloud | sonarqube

|   |
|---|
| <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>               |
| <div>String literals should not be duplicated</div> <div> Code Smell</div>                               |