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

Cryptographic keys should be robust

Analyze your code

Vulnerability | Critical | cwe privacy owasp

Most of cryptographic systems require a sufficient key size to be robust against brute-force attacks.

NIST recommendations will be checked for these use-cases:

Digital Signature Generation and **Verification:**

- p ≥ 2048 AND q ≥ 224 for DSA (`p` is key length and `q` the modulus length)
- n ≥ 2048 for RSA (`n` is the key length)

Key Agreement:

- p ≥ 2048 AND q ≥ 224 for DH and MQV
- n ≥ 224 for ECDH and ECMQV (Examples: `secp192r1` is a non-compliant curve (`n` < 224) but `secp224k1` is compliant (`n` >= 224))

Symmetric keys:

- key length ≥ 128 bits

This rule will not raise issues for ciphers that are considered weak (no matter the key size) like `DES`, `Blowfish`.

Noncompliant Code Example

```
val keyPairGen1 = KeyPairGenerator.getInstance("RSA")
keyPairGen1.initialize(1024) // Noncompliant

val keyPairGen5 = KeyPairGenerator.getInstance("EC")
val ecSpec1 = ECGenParameterSpec("secp112r1") // Noncomp
keyPairGen5.initialize(ecSpec1)

val keyGen1 = KeyGenerator.getInstance("AES")
keyGen1.init(64) // Noncompliant
```

Compliant Solution

```
val keyPairGen6 = KeyPairGenerator.getInstance("RSA")
keyPairGen6.initialize(2048) // Compliant

val keyPairGen5 = KeyPairGenerator.getInstance("EC")
val ecSpec1 = ECGenParameterSpec("secp256r1") // Complia
keyPairGen5.initialize(ecSpec1)

val keyGen2 = KeyGenerator.getInstance("AES")
keyGen2.init(128) // Compliant
```

See

- OWASP Top 10 2021 Category A2 - Cryptographic Failures
- OWASP Top 10 2017 Category A3 - Sensitive Data Exposure
- OWASP Top 10 2017 Category A6 - Security Misconfiguration
- Mobile AppSec Verification Standard - Cryptography Requirements
- OWASP Mobile Top 10 2016 Category M5 - Insufficient Cryptography
- NIST 800-131A - Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths
- MITRE, CWE-326 - Inadequate Encryption Strength

"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

Available In:

sonarlint | sonarcloud | sonarqube