

CloudFormation

COBOL COBOL

C++

C# C#

∃ CSS

⋈ Flex

GO Go

⊎ HTML

Java JavaScript

Kotlin

Kubernetes

Objective C

PHP

PL/I

PL/SQL PL/SQL

Python

RPG RPG

Ruby

Scala

Swift

Terraform

Text

TS TypeScript

T-SQL

VB VB.NET

VB6 VB6

XML XML



Kotlin static code analysis

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

Hard-coded credentials are securitysensitive 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

Weak SSL/TLS protocols should not be used

Vulnerability

"SecureRandom" seeds should not be

Vulnerability

G Vulnerability

Cipher Block Chaining IVs should be unpredictable

Vulnerability

predictable

Hashes should include an unpredictable salt

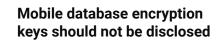
■ Vulnerability

Regular expressions should be syntactically valid

📆 Bug

"runFinalizersOnExit" should not be called

📆 Bug



Tags

Analyze your code

Code Smell (56)

Search by name...

6 Vulnerability ♦ Major ② ■ cwe owasp android

Storing data locally is a common task for mobile applications. There are many convenient solutions that allow storing data persistently, for example SQLiteDatabase and Realm. These systems can be initialized with a secret key in order to store the data encrypted.

The encryption key is meant to stay secret and should not be hard-coded in the application as it would mean that:

- All user would use the same encryption key.
- The encryption key would be known by anyone who as access to the source code or the application binary code.
- Data stored encrypted in the database would not be protected.

There are different approaches how the key can be provided to encrypt and decrypt the database. One of the most convinient way to is to rely on EncryptedSharedPreferences to store encryption keys. It can also be provided dynamically by the user of the application or fetched from a remote server

Noncompliant Code Example

SQLCipher

val key = "gb09ym9ydoolp3w886d0tciczj6ve9kszqd65u7d12604
val db = SQLiteDatabase.openOrCreateDatabase("test.db",

Realm

val key = "gb09ym9ydoolp3w886d0tciczj6ve9kszqd65u7d12604
val config = RealmConfiguration.Builder()
 .encryptionKey(key.toByteArray()) // Noncompliant
 .build()
val realm = Realm.getInstance(config)

Compliant Solution

SQLCipher

val db = SQLiteDatabase.openOrCreateDatabase("test.db",

Realm

val config = RealmConfiguration.Builder()
 .encryptionKey(getKey())
 .build()
val realm = Realm.getInstance(config)

See

- OWASP Top 10 2021 Category A2 Cryptographic Failures
- OWASP Top 10 2021 Category A4 Insecure Design
- Mobile AppSec Verification Standard Data Storage and Privacy Requirements
- OWASP Mobile Top 10 2016 Category M2 Insecure Data Storage

"ScheduledThreadPoolExecutor" should not have 0 core threads Rug Bug Jump statements should not occur in "finally" blocks Rug Bug Using clear-text protocols is securitysensitive Security Hotspot Accessing Android external storage is security-sensitive Security Hotspot Receiving intents is security-sensitive Security Hotspot Broadcasting intents is securitysensitive Security Hotspot Using weak hashing algorithms is security-sensitive Security Hotspot Using pseudorandom number generators (PRNGs) is securitysensitive 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

- OWASP Top 10 2017 Category A3 Sensitive Data Exposure
- OWASP Top 10 2017 Category A6 Security Misconfiguration
- MITRE, CWE-311 Missing Encryption of Sensitive Data
- MITRE, CWE-321 Use of Hard-coded Cryptographic Key

Available In:

sonarlint ⊕ | sonarcloud ♦ | sonarqube

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved. Privacy Policy