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

Using biometric authentication without a cryptographic solution is security-sensitive

Analyze your code

Security Hotspot | Major | cwe owasp

Android comes with Android KeyStore, a secure container for storing key materials. It's possible to define certain keys to be unlocked when users authenticate using biometric credentials. This way, even if the application process is compromised, the attacker cannot access keys, as presence of the authorized user is required.

These keys can be used, to encrypt, sign or create a message authentication code (MAC) as proof that the authentication result has not been tampered with. This protection defeats the scenario where an attacker with physical access to the device would try to hook into the application process and call the `onAuthenticationSucceeded` method directly. Therefore he would be unable to extract the sensitive data or to perform the critical operations protected by the biometric authentication.

Ask Yourself Whether

The application contains:

- Cryptographic keys / sensitive information that need to be protected using biometric authentication.

There is a risk if you answered yes to this question.

Recommended Secure Coding Practices

It's recommended to tie the biometric authentication to a cryptographic operation by using a `CryptoObject` during authentication.

Noncompliant Code Example

A `CryptoObject`is not used during authentication:

```
// ...
val biometricPrompt: BiometricPrompt = BiometricPrompt(a
// ...
biometricPrompt.authenticate(promptInfo) // Noncompliant
```

Compliant Solution

A `CryptoObject` is used during authentication:

```
// ...
val biometricPrompt: BiometricPrompt = BiometricPrompt(a
// ...
biometricPrompt.authenticate(promptInfo, BiometricPrompt
```

See

- OWASP Top 10 2021 Category A7 - Identification and Authentication Failures
- developer.android.com - Use a cryptographic solution that depends on authentication
- OWASP Mobile Top 10 Category M4 - Insecure Authentication
- OWASP MASVS - Authentication and Session Management Requirements

- MITRE, CWE-287 - Improper Authentication

Available In:

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