sonar RULES

Products

- 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

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

Using unencrypted files in mobile applications is security-sensitive

Analyze your code

Security Hotspot   Major   cwe owasp android

Storing files locally is a common task for mobile applications. Files that are stored unencrypted can be read out and modified by an attacker with physical access to the device. Access to sensitive data can be harmful for the user of the application, for example when the device gets stolen.

Ask Yourself Whether

- The file contains sensitive data that could cause harm when leaked.

There is a risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

It's recommended to password-encrypt local files that contain sensitive information. The class EncryptedFile can be used to easily encrypt files.

Sensitive Code Example

```
val targetFile = File(activity.filesDir, "data.txt")
targetFile.writeText(fileContent)  // Sensitive
```

Compliant Solution

```
val mainKey = MasterKeys.getOrCreate(MasterKeys.AES256_G

val encryptedFile = EncryptedFile.Builder(
    File(activity.filesDir, "data.txt"),
    activity,
    mainKey,
    EncryptedFile.FileEncryptionScheme.AES256_GCM_HKDF_4
).build()

encryptedFile.openFileOutput().apply {
    write(fileContent)
    flush()
    close()
}
```

See

- 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
- 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

Available In:

sonarcloud sonarqube