



Kotlin static code analysis

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

₩ Bug 17

Tags

All rules 98	rability 10
Hard-coded credentials are sensitive	security-
Security Hotspot	
Cipher algorithms should be	e robust
★ Vulnerability	
Encryption algorithms shou with secure mode and padd scheme	
C Vulnerability	
Server hostnames should be during SSL/TLS connections	
★ Vulnerability	
Server certificates should be during SSL/TLS connections	
☆ Vulnerability	
Cryptographic keys should be Vulnerability	oe robust
Weak SSL/TLS protocols sh	ould not
⋒ Vulnerability	
"SecureRandom" seeds sho predictable	uld not be
☆ Vulnerability	
Cipher Block Chaining IVs s unpredictable	hould be
G Vulnerability	
Hashes should include an unpredictable salt	
6 Vulnerability	
Regular expressions should syntactically valid	be
Rug	
"runFinalizersOnExit" should called	l not be
👚 Bug	

~	Search by name	Q

Code Smell 56

Security Hotspot (15)

"ScheduledThreadPoolExecutor" should not have 0 core threads

📆 Bug

Jump statements should not occur in "finally" blocks

📆 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

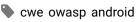
Code Smell

Cognitive Complexity of functions should not be too high

Code Smell

Using unencrypted files in mobile applications is security-sensitive

Analyze your code



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

© 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