



COBOL COBOL

C# C#

g css

X Flex

-co Go

⊎ HTML

Java Java Java Script

Kotlin

Kubernetes

Objective C

PHP

PL/I

PL/SQL

Python

RPG RPG

Ruby

Swift

Terraform

Scala

■ Text

Ts TypeScript

T-SQL

VB VB.NET

VB6 VB6

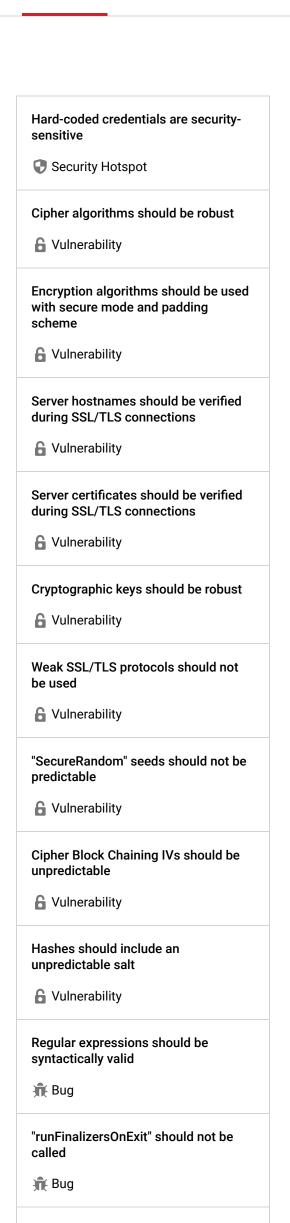
XML XML



Kotlin static code analysis

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

Tags



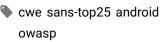


Accessing Android external storage is security-sensitive

Analyze your code

Security Hotspot





Search by name...

Storing data locally is a common task for mobile applications. Such data includes files among other things. One convenient way to store files is to use the external file storage which usually offers a larger amount of disc space compared to internal storage.

Files created on the external storage are globally readable and writable.

Therefore, a malicious application having the permissions

WRITE_EXTERNAL_STORAGE or READ_EXTERNAL_STORAGE could try to read sensitive information from the files that other applications have stored on the external storage.

External storage can also be removed by the user (e.g when based on SD card) making the files unavailable to the application.

Ask Yourself Whether

Your application uses external storage to:

- store files that contain sensitive data.
- store files that are not meant to be shared with other application.
- store files that are critical for the application to work.

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

Recommended Secure Coding Practices

- Use internal storage whenever possible as the system prevents other apps from accessing this location.
- Only use external storage if you need to share non-sensitive files with other applications.
- If your application has to use the external storage to store sensitive data, make sure it encrypts the files using EncryptedFile.
- Data coming from external storage should always be considered untrusted and should be validated.
- As some external storage can be removed, make sure to never store files on it that are critical for the usability of your application.

Sensitive Code Example

```
import android.content.Context

class AccessExternalFiles {
    fun accessFiles(Context context) {
        context.getExternalFilesDir(null) // Sensitive
    }
}
```

Compliant Solution

```
import android.content.Context
import android.os.Environment
class AccessExternalFiles {
```

"ScheduledThreadPoolExecutor" should not have 0 core threads Rug 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

```
fun accessFiles(Context context) {
        context.getFilesDir()
    }
}
```

See

- OWASP Top 10 2021 Category A4 Insecure Design
- Android Security tips on external file storage
- Mobile AppSec Verification Standard Data Storage and Privacy Requirements
- OWASP Mobile Top 10 2016 Category M2 Insecure Data Storage
- MITRE, CWE-312 Cleartext Storage of Sensitive Information
- SANS Top 25 Risky Resource Management
- SANS Top 25 Porous Defenses

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