Secrets
ABAP
Apex
C
C++
CloudFormation
COBOL
C#
CSS
Flex
Go
HTML
Java
JavaScript
Kotlin
Objective C
PHP
PL/I
PL/SQL
Python
RPG
Ruby
Scala
Swift
Terraform
Text
TypeScript
T-SQL
VB.NET
VB6
XML

Java static code analysis

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

All rules 632 | Vulnerability 53 | Bug 154 | Security Hotspot 36 | Code Smell 389 | Quick Fix 42

Tags | Search by name...

Code Smell

Redundant casts should not be used
Code Smell

"@Deprecated" code should not be used
Code Smell

"toString()" should never be called on a String object
Code Smell

Annotation repetitions should not be wrapped
Code Smell

Multiple variables should not be declared on the same line
Code Smell

Strings should not be concatenated using '+' in a loop
Code Smell

Maps with keys that are enum values should be replaced with EnumMap
Code Smell

Lambdas should be replaced with method references
Code Smell

Parentheses should be removed from a single lambda input parameter when its type is inferred
Code Smell

Abstract classes without fields should be converted to interfaces
Code Smell

Lambdas containing only one statement should not nest this statement in a block
Code Smell

Authorizing non-authenticated users to use keys in the Android KeyStore is security-sensitive

Analyze your code

Security Hotspot | Major | cwe owasp android

Android KeyStore is a secure container for storing key materials, in particular it prevents key materials extraction, i.e. when the application process is compromised, the attacker cannot extract keys but may still be able to use them. It's possible to enable an Android security feature, user authentication, to restrict usage of keys to only authenticated users. The lock screen has to be unlocked with defined credentials (pattern/PIN/password, biometric).

Ask Yourself Whether

- The application requires prohibiting the use of keys in case of compromise of the application process.
- The key material is used in the context of a highly sensitive application like a e-banking mobile app.

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

Recommended Secure Coding Practices

It's recommended to enable user authentication (by setting `setUserAuthenticationRequired` to `true` during key generation) to use keys for a limited duration of time (by setting appropriate values to `setUserAuthenticationValidityDurationSeconds`), after which the user must re-authenticate.

Noncompliant Code Example

Any user can use the key:

```
KeyGenerator keyGenerator = KeyGenerator.getInstance(KeyProp

KeyGenParameterSpec builder = new KeyGenParameterSpec.Builde
    .setBlockModes(KeyProperties.BLOCK_MODE_GCM)
    .setEncryptionPaddings(KeyProperties.ENCRYPTION_PADDING_
    .build();

keyGenerator.init(builder);
```

Compliant Solution

The use of the key is limited to authenticated users (for a duration of time defined to 60 seconds):

```
KeyGenerator keyGenerator = KeyGenerator.getInstance(KeyProp

KeyGenParameterSpec builder = new KeyGenParameterSpec.Builde
    .setBlockModes(KeyProperties.BLOCK_MODE_GCM)
    .setEncryptionPaddings(KeyProperties.ENCRYPTION_PADDING_
    .setUserAuthenticationRequired(true)
    .setUserAuthenticationParameters (60, KeyProperties.AUTH
    .build();
```

"Collections.EMPTY_LIST", "EMPTY_MAP", and "EMPTY_SET" should not be used

Code Smell

Local variables should not be declared and then immediately returned or thrown

Code Smell

Unused local variables should be removed

Code Smell

Private fields only used as local

```
keyGenerator.init(builder)
```

See

- OWASP Top 10 2021 Category A4 - Insecure Design
- developer.android.com - Android keystore system
- developer.android.com - Require user authentication for key use
- Mobile AppSec Verification Standard - Authentication and Session Management Requirements
- OWASP Mobile Top 10 2016 Category M4 - Insecure Authentication
- MITRE, CWE-522 - Insufficiently Protected Credentials

Available In:

sonarcloud | sonarqube