

-  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

Receiving intents is security-sensitive

Analyze your code

Security Hotspot

Critical ?

cwe owasp sans-top25 android

Android applications can receive broadcasts from the system or other applications. Receiving intents is security-sensitive. For example, it has led in the past to the following vulnerabilities:

- [CVE-2019-1677](#)
- [CVE-2015-1275](#)

Receivers can be declared in the manifest or in the code to make them context specific. If the receiver is declared in the manifest Android will start the application if it is not already running once a matching broadcast is received. The receiver is an entry point into the application.

Other applications can send potentially malicious broadcasts, so it is important to consider broadcasts as untrusted and to limit the applications that can send broadcasts to the receiver.

Permissions can be specified to restrict broadcasts to authorized applications. Restrictions can be enforced by both the sender and receiver of a broadcast. If permissions are specified when registering a broadcast receiver, then only broadcasters who were granted this permission can send a message to the receiver.

This rule raises an issue when a receiver is registered without specifying any "broadcast permission".

Ask Yourself Whether

- The data extracted from intents is not sanitized.
- Intents broadcast is not restricted.

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

Recommended Secure Coding Practices











Restrict the access to broadcasted intents. See [Android documentation](#) for more information.

Sensitive Code Example

```
import android.content.BroadcastReceiver
import android.content.Context
import android.content.IntentFilter
import android.os.Build
import android.os.Handler
import androidx.annotation.RequiresApi

class MyIntentReceiver {
    @RequiresApi(api = Build.VERSION_CODES.O)
    fun register(
        context: Context, receiver: BroadcastReceiver?,
        filter: IntentFilter?,
        scheduler: Handler?,
        flags: Int
    ) {
        context.registerReceiver(receiver, filter) // Se
        context.registerReceiver(receiver, filter, flags

        // Broadcasting intent with "null" for broadcast
```

<div>"ScheduledThreadPoolExecutor" should not have 0 core threads</div> <div> Bug</div>
<div>Jump statements should not occur in "finally" blocks</div> <div> Bug</div>
<div>Using clear-text protocols is security-sensitive</div> <div> Security Hotspot</div>
<div>Accessing Android external storage is security-sensitive</div> <div> Security Hotspot</div>
<div>Receiving intents is security-sensitive</div> <div> Security Hotspot</div>
<div>Broadcasting intents is security-sensitive</div> <div> Security Hotspot</div>
<div>Using weak hashing algorithms is security-sensitive</div> <div> Security Hotspot</div>
<div>Using pseudorandom number generators (PRNGs) is security-sensitive</div> <div> Security Hotspot</div>
<div>Empty lines should not be tested with regex MULTILINE flag</div> <div> Code Smell</div>
<div>Cognitive Complexity of functions should not be too high</div> <div> Code Smell</div>

```
        context.registerReceiver(receiver, filter, null,
                                context.registerReceiver(receiver, filter, null,
                                }
    }
```

Compliant Solution

```
import android.content.BroadcastReceiver
import android.content.Context
import android.content.IntentFilter
import android.os.Build
import android.os.Handler
import androidx.annotation.RequiresApi

class MyIntentReceiver {
    @RequiresApi(api = Build.VERSION_CODES.O)
    fun register(
        context: Context, receiver: BroadcastReceiver?,
        filter: IntentFilter?,
        broadcastPermission: String?,
        scheduler: Handler?,
        flags: Int
    ) {
        context.registerReceiver(receiver, filter, broad
        context.registerReceiver(receiver, filter, broad
    }
}
```

See

- [Mobile AppSec Verification Standard](#) - Platform Interaction Requirements
- [OWASP Mobile Top 10 2016 Category M1](#) - Improper Platform Usage
- [MITRE, CWE-925](#) - Improper Verification of Intent by Broadcast Receiver
- [MITRE, CWE-926](#) - Improper Export of Android Application Components
- [SANS Top 25](#) - Insecure Interaction Between Components
- [Android documentation](#) - Broadcast Overview - Security considerations and best practices

Available In:

sonarcloud



|

sonarqube

