




 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 rules632

Vulnerability53

Bug154

Security Hotspot36

Code Smell389

Quick Fix42

Tags ▾

Search by name... 🔍

Regex alternatives should not be redundant

Bug

Alternatives in regular expressions should be grouped when used with anchors

Bug

AssertJ methods setting the assertion context should come before an assertion

Bug

AssertJ configuration should be applied

Bug

JUnit5 test classes and methods should not be silently ignored

Bug

"ThreadLocal" variables should be cleaned up when no longer used

Bug

Strings and Boxed types should be compared using "equals()"

Bug

InputStream.read() implementation should not return a signed byte

Bug

"compareTo" should not be overloaded

Bug

"iterator" should not return "this"

Bug

Map values should not be replaced unconditionally

Bug

Week Year ("YYYY") should not be used for date formatting

Receiving intents is security-sensitive

Analyze your code

Security HotspotCritical🔍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 android.support.annotation.RequiresApi;

public class MyIntentReceiver {

    @RequiresApi(api = Build.VERSION_CODES.O)
    public void register(Context context, BroadcastReceiver
                        IntentFilter filter,
                        String broadcastPermission,
                        Handler scheduler,
```

https://rules.sonarsource.com/java/RSPEC-5322

1/2

 Bug
Exceptions should not be created without being thrown
 Bug
Collection sizes and array length comparisons should make sense
 Bug
Consumed Stream pipelines should not be reused
 Bug
Intermediate Stream methods should not be left unused

```
        int flags) {
    context.registerReceiver(receiver, filter); // Sensi
    context.registerReceiver(receiver, filter, flags); /

    // Broadcasting intent with "null" for broadcastPerm
    context.registerReceiver(receiver, filter, null, sch
    context.registerReceiver(receiver, filter, null, sch

    }
}
```

Compliant Solution

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

public class MyIntentReceiver {

    @RequiresApi(api = Build.VERSION_CODES.O)
    public void register(Context context, BroadcastReceiver
                        IntentFilter filter,
                        String broadcastPermission,
                        Handler scheduler,
                        int flags) {

        context.registerReceiver(receiver, filter, broadcast
        context.registerReceiver(receiver, filter, broadcast

    }
}
```

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:  
 | 