

- 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...

Components should not be vulnerable to intent redirection

Vulnerability

XML parsers should not allow inclusion of arbitrary files

Vulnerability

HTTP responses should not be vulnerable to session fixation

Vulnerability

Extracting archives should not lead to zip slip vulnerabilities

Vulnerability

Dynamic code execution should not be vulnerable to injection attacks

Vulnerability

NoSQL operations should not be vulnerable to injection attacks

Vulnerability

HTTP request redirections should not be open to forging attacks

Vulnerability

Deserialization should not be vulnerable to injection attacks

Vulnerability

Endpoints should not be vulnerable to reflected cross-site scripting (XSS) attacks

Vulnerability

Database queries should not be vulnerable to injection attacks

Vulnerability

XML parsers should not be vulnerable to XXE attacks

Vulnerability

Components should not be vulnerable to intent redirection

Analyze your code

Vulnerability Blocker injection android

Intent redirections are intents forwarded by a component to another component. It can lead to vulnerabilities such as leak of information when an exported component of the mobile app, like an activity, doesn't validate an untrusted intent used to start a component, ie to perform the intent redirection.

Noncompliant Code Example

A component activity is exported (in this case using an intent-filter) allowing it to be launched by other mobile applications:

```
<activity android:name=".Noncompliant">
  <intent-filter>
    <action android:name="noncompliantaction" />
  </intent-filter>
</activity>
```

Then this activity retrieves the embedded untrusted intent used to start an arbitrary component:

```
public class Noncompliant extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // The intent used to start this exported component
        Intent intent = getIntent();

        // extract the embedded Intent
        Intent forward = (Intent) intent.getParcelableExtra(

        // redirect the embedded Intent
        startActivity(forward); // Noncompliant
    }
}
```

Compliant Solution


If it's not needed to make visible this component to other apps, do not export it:

```
<activity android:name=".Noncompliant" android:exported="false">
  <intent-filter>
    <action android:name="noncompliantaction" />
  </intent-filter>
</activity>
```

It's also possible to validate the intent to be sure it's the expected one:

```
public class Noncompliant extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
```


A secure password should be used when connecting to a database

 Vulnerability


XPath expressions should not be vulnerable to injection attacks

 Vulnerability

I/O function calls should not be vulnerable to path injection attacks

 Vulnerability

LDAP queries should not be vulnerable to injection attacks

 Vulnerability

OS commands should not be

```
// The intent used to start this exported component
Intent intent = getIntent();

// extract the embedded Intent
Intent forward = (Intent) intent.getParcelableExtra(

ComponentName name = forward.resolveActivity(getPack
if (name.getPackageName().equals("package") &&
    name.getClassName().equals("nonsensitiveclas
    // redirect the embedded Intent
    startActivity(forward);
}
}
}
```

See

- [support.google.com](#) - Remediation for Intent Redirection Vulnerability
- [Mobile AppSec Verification Standard](#) - Platform Interaction Requirements
- [OWASP Mobile Top 10 2016 Category M1](#) - Improper Platform Usage
- [MITRE, CWE-20](#) - Improper Input Validation

Available In:

sonarcloud  | **sonarqube**  Developer Edition