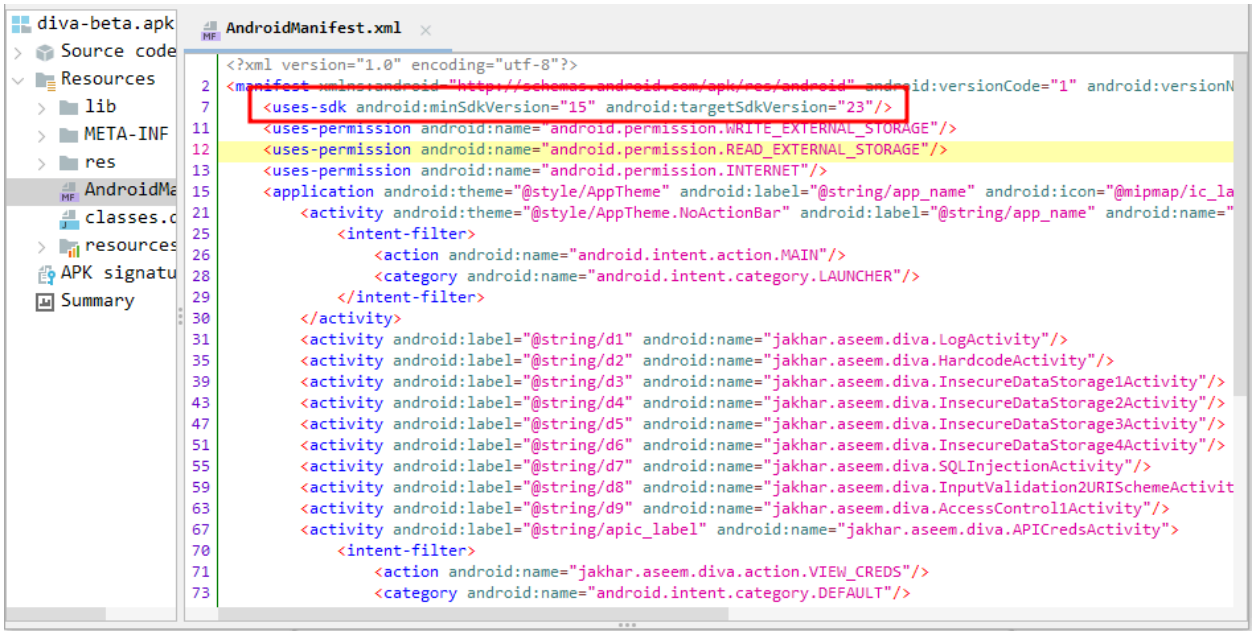
**Vulnerability Assessment Report for diva app**

Submitted by :- Sauda Momin

## 1. Minimum SDK Version is Too Low

**Description:**
The application sets a very low minSdkVersion (15, Android 4.0.3), allowing the app to run on outdated Android versions with numerous security vulnerabilities.




**Root Cause:**
The android:minSdkVersion in the AndroidManifest.xml is set to an old API level, making the app compatible with legacy devices but exposing it to known security issues.

**Impact:**

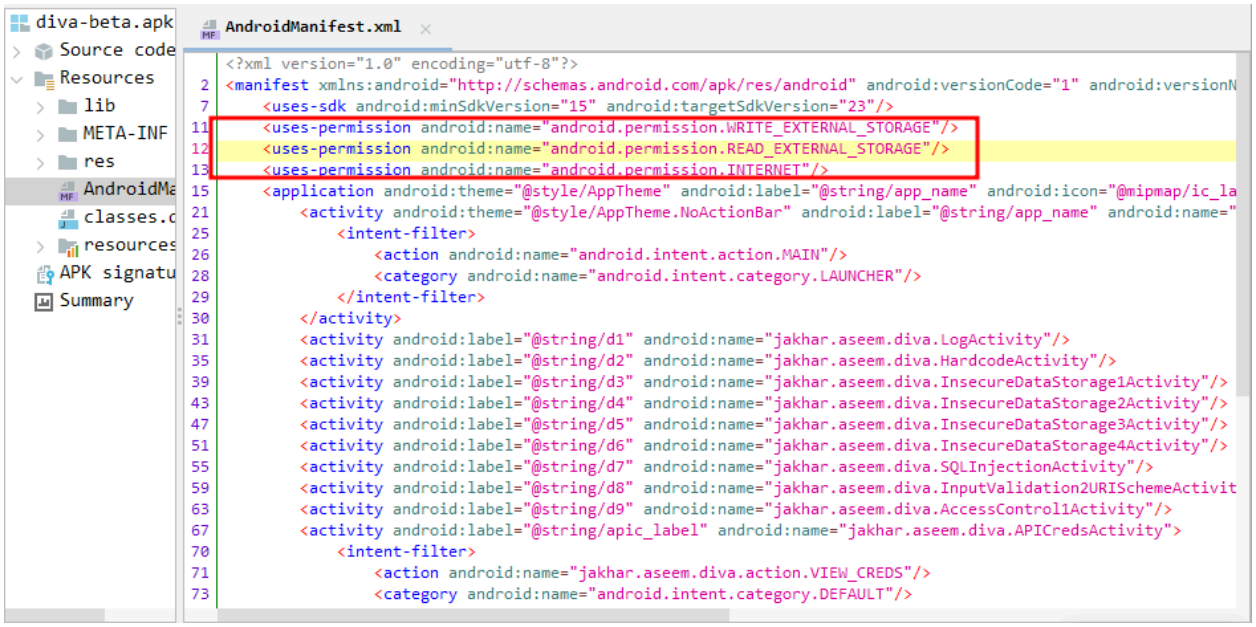
- Devices running old Android versions may lack modern security features and patches.
- Increases risk of data leaks, privilege escalation, and exploitation of known OS vulnerabilities.

**Mitigation:**

- Raise the minSdkVersion to 21 or higher (Android 5.0+) to ensure baseline security features.
- Test the app for compatibility with the new minimum SDK to avoid runtime issues.

## 2. Sensitive Storage Permissions

**Description:**
The application requests sensitive storage permissions (e.g., READ_EXTERNAL_STORAGE or WRITE_EXTERNAL_STORAGE) that provide full access to the device's storage.




**Root Cause:**
The app declares storage permissions in the manifest, granting it access to all files on external storage, without restricting use to necessary files only.

**Impact:**

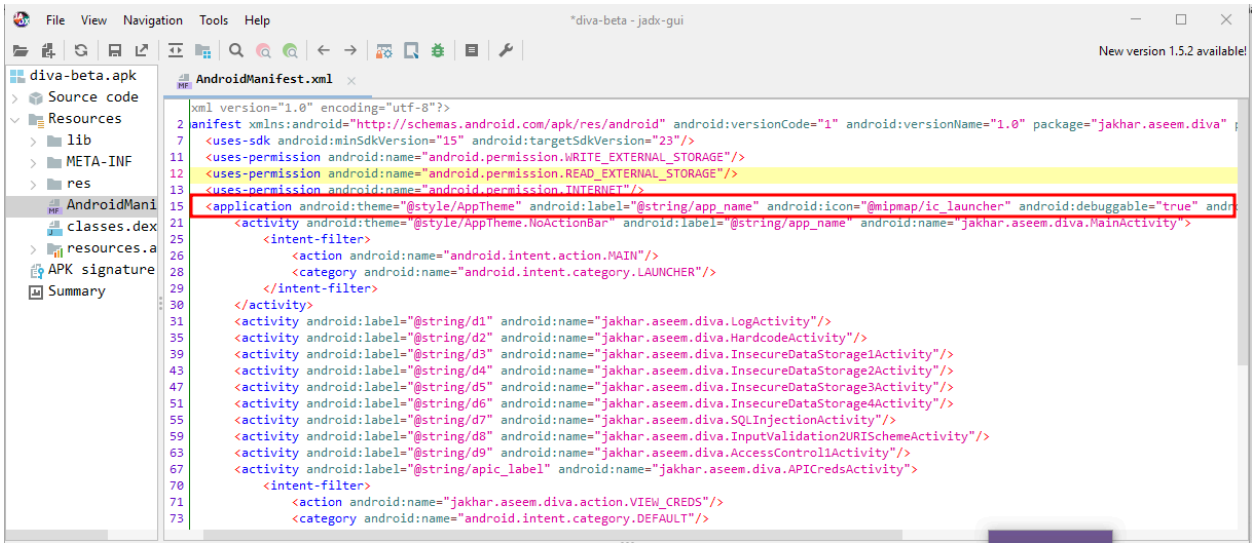
- Any files stored on external storage, including user data or sensitive app files, could be exposed, modified, or deleted by other apps.
- Increases the risk of data leakage, tampering, and unauthorized access.

**Mitigation:**

- Remove sensitive storage permissions if they are not required
- Use internal app storage or scoped storage APIs (Android 10/API 29+) for sensitive files.
- Restrict file access to app-private directories and avoid storing sensitive data in shared locations.

## 3. Debuggable App Enabled

**Description:**
The application is built with debugging enabled (android:debuggable="true"), allowing external tools to attach a debugger to the app.




**Root Cause:**
The android:debuggable attribute in the AndroidManifest.xml is set to true, either manually or due to a debug build configuration, exposing the app in production.

**Impact:**

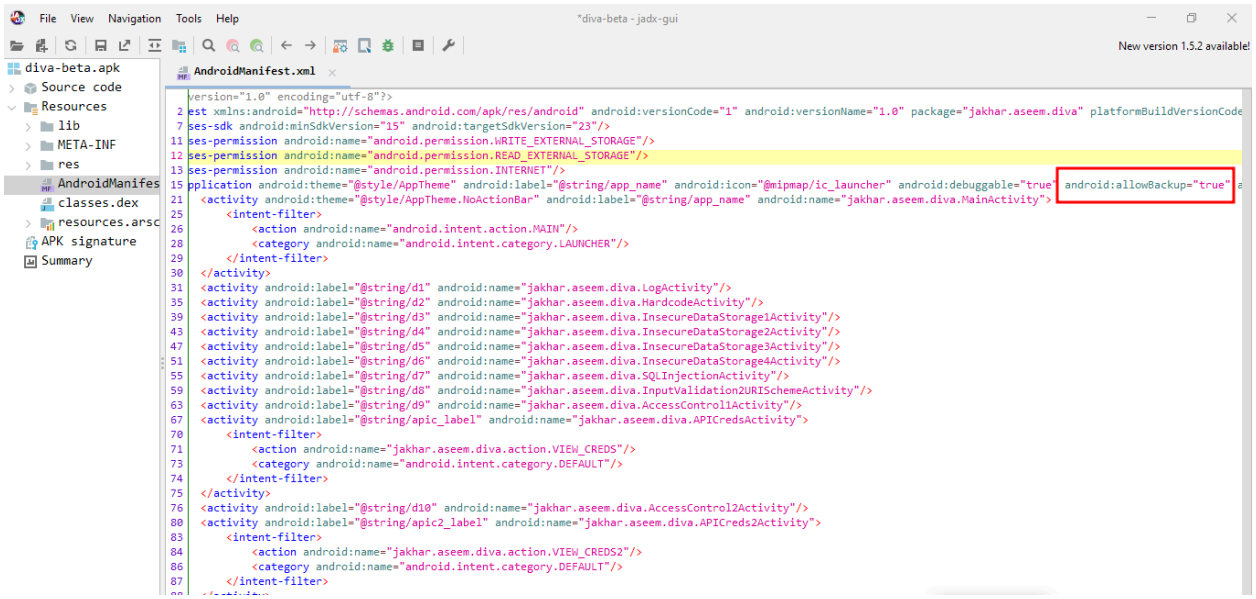
- Attackers can reverse engineer, inspect memory, or manipulate app behavior.
- Sensitive data such as credentials or encryption keys can be extracted.
- Increases the risk of security bypasses and unauthorized actions.

**Mitigation:**

- Set android:debuggable="false" for all release builds.
- Maintain separate debug and release build variants.
- Verify release APKs using aapt dump badging <app.apk> to confirm debuggable is disabled.

## 4. App Backup Allowed

**Description:**
The application allows backup of its data (android:allowBackup="true"), enabling Android's backup mechanisms to store and restore app data.




**Root Cause:**
The android:allowBackup attribute in the AndroidManifest.xml is set to true, allowing any user or attacker with device or backup access to copy app data.

**Impact:**

- Sensitive data stored by the app can be extracted from backups.
- Increases risk of data leakage, privacy violations, and unauthorized access to user information.

**Mitigation:**

- Set android:allowBackup="false" for release builds.
- Review sensitive data stored in the app and ensure it is protected even if backups are enabled.
- Use **encryption** for any critical stored data if backups are necessary.

## 5. Exported Content Provider

**Description:**

The application's ContentProvider is exported (android:exported="true"), allowing other apps on the device to access or modify its data.


**Root Cause:**
The android:exported attribute in the AndroidManifest.xml is set to true, and no permissions are enforced, making the provider publicly accessible.

**Impact:**

- Other apps can read, modify, or delete data in the content provider.
- May lead to data leakage, unauthorized modifications, or app logic manipulation.

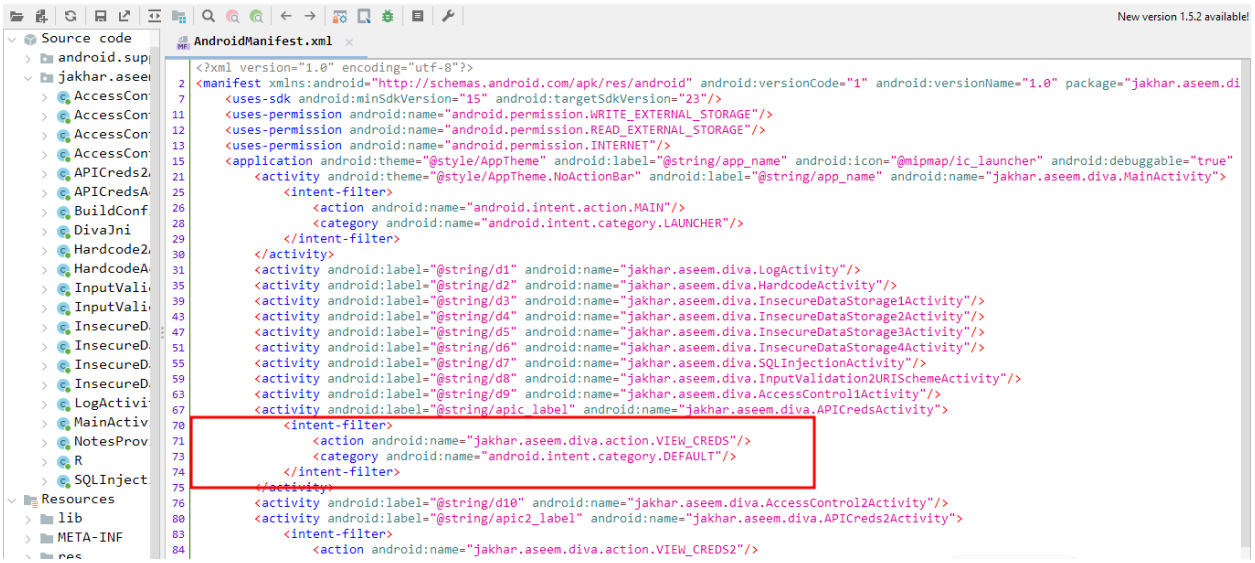
**Mitigation:**

- Set android:exported="false" if the provider is not intended for external access.
- If external access is required, enforce **read/write permissions** using android:readPermission and android:writePermission.
- Validate all data received via content URIs before processing.

## 6. Unprotected Activity with Intent Filter

**Description:**

The application exposes an activity via an intent filter without any protection, allowing other apps to send intents to it.




**Root Cause:**

The activity is declared in the AndroidManifest.xml with an intent-filter but without any permission restrictions, making it publicly accessible.

**Impact:**

- Malicious apps can launch the activity and potentially access sensitive data like API keys or user information.
- Increases the risk of data theft or unauthorized app behavior manipulation.

**Mitigation:**

- Protect the activity by adding a custom permission in the manifest: android:permission="your.app.PERMISSION"
- Only allow trusted apps or components to invoke the activity.
- Validate all inputs received via intents before processing.