



XML

Background

Android uses XML files to store static data, meaning that the data does not change through execution of the app. For example, translated strings (in languages such as English, Spanish, or Arabic) are stored in XML files for multi-language support. APK files are compressed files that contain many XML and Java files, making up the application. Note that anyone could unzip the APK and read the data that was saved in XML.

Many developers store important information in XML, such as an Ads number or a Google Maps API key. That is precarious, because someone who reverse engineers the app could read all the keys. The attacker could then use the stolen API key to obtain access to paid services, possibly running up high charges on the developer's account.

Activity Instructions

We will show how this is a problem by doing the following:

- i. Build an app that displays Google Maps with the API key saved in XML
- ii. Extract the key from the APK
- iii. Learn the best place to store the key so that this does not happen

1. Project Creation

- a. Follow the screens below to create a new project:

Name the project "AndroidGoogleMAP". Take note of the package name.



Create New Project

New Project

Android Studio

Configure your new project

Application name:

Company Domain:

Package name: [Edit](#)

Project location: ...

[Cancel](#) [Previous](#) [Next](#) [Finish](#)

Create New Project

Target Android Devices

Select the form factors your app will run on

Different platforms may require separate SDKs

☒ Phone and Tablet

Minimum SDK: [Help me choose](#)

Lower API levels target more devices, but have fewer features available.
By targeting API 16 and later, your app will run on approximately **96.7%** of the devices that are active on the Google Play Store.
Stats load failed. Value may be out of date.

☐ Wear

Minimum SDK:

☐ TV

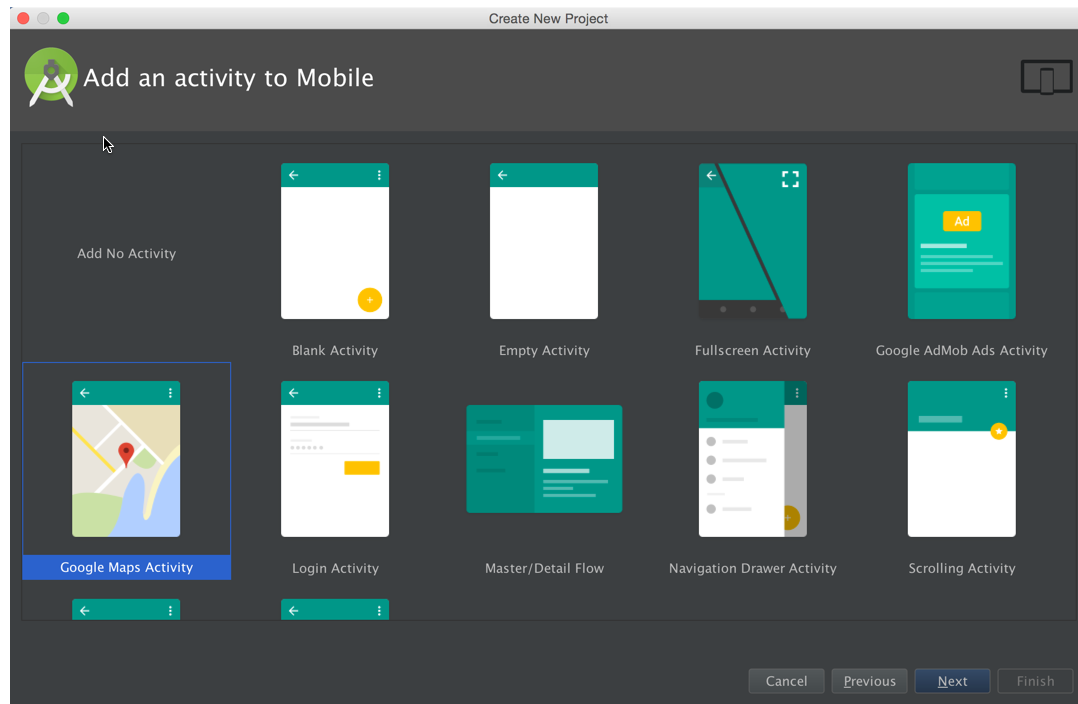
Minimum SDK:

☐ Android Auto

☐ Glass (Not Available)

Minimum SDK:

[Previous](#) [Next](#) [Cancel](#) [Finish](#)



Select activity of type “Google Maps Activity”.

2. Modify the XML File

See the app files under “app/res/values/”. One of them is **strings.xml** file. It has a key named “google_maps_key”. We must create the Google Maps key for this app from the Google developer console by following the instructions in this file. After the key is created, replace the placeholder “YOUR_KEY_HERE” with the generated key in quote marks.

```
MapsActivity.java x google_maps_api.xml x
resources
<resources>
  <!--
    TODO: Before you run your application, you need a Google Maps API key.

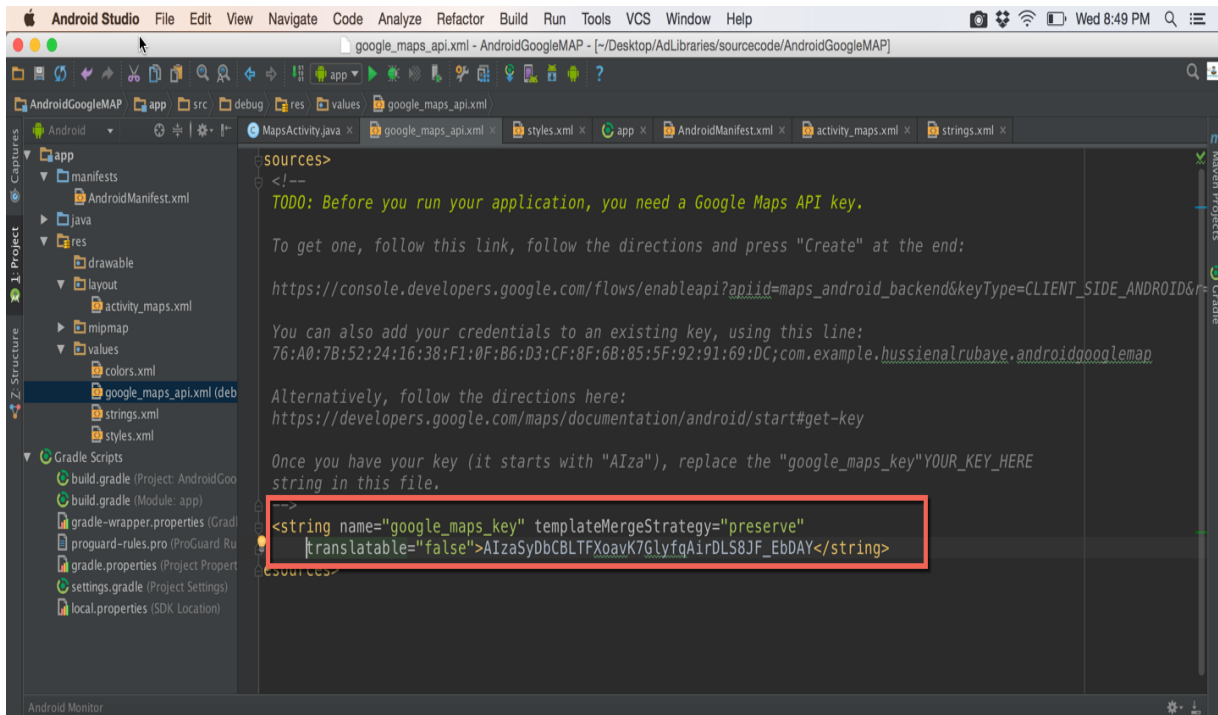
    To get one, follow this link, follow the directions and press "Create" at the end:
    https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend&keyType=CLIENT_SIDE_ANDROID&r=

    You can also add your credentials to an existing key, using this line:
    BB:76:68:7A:5A:F2:86:F3:4C:7D:5B:93:0F:9D:11:6D:ED:F1:66:6F;com.example.android.googlemap

    Alternatively, follow the directions here:
    https://developers.google.com/maps/documentation/android/start#get-key

    Once you have your key (it starts with "AIza"), replace the "google_maps_key"
    string in this file.
  -->
  <string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">YOUR_KEY_HERE</string>
</resources>
```

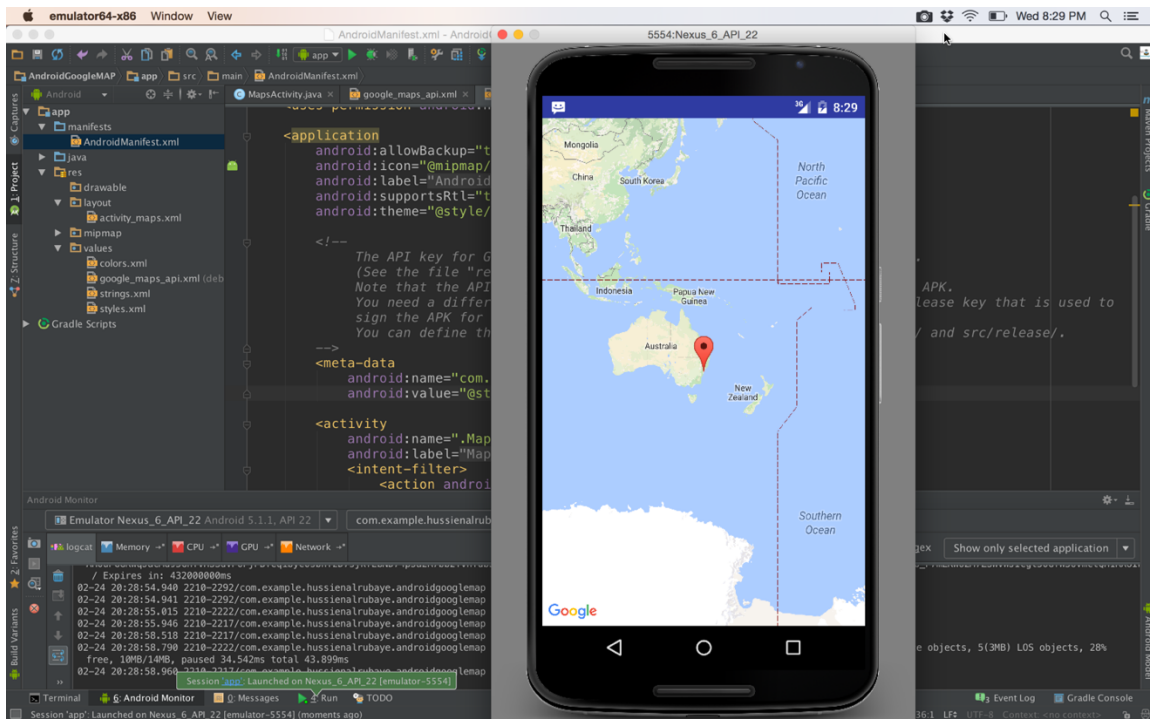
After replacing, you should see something like this:



Exploitation Instructions

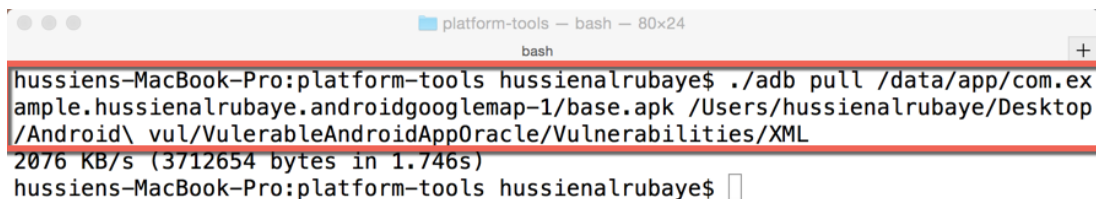
We shall see for ourselves how we could be able to retrieve the API key from the APK.

1. Run the app. See that we have Google Maps in our app.

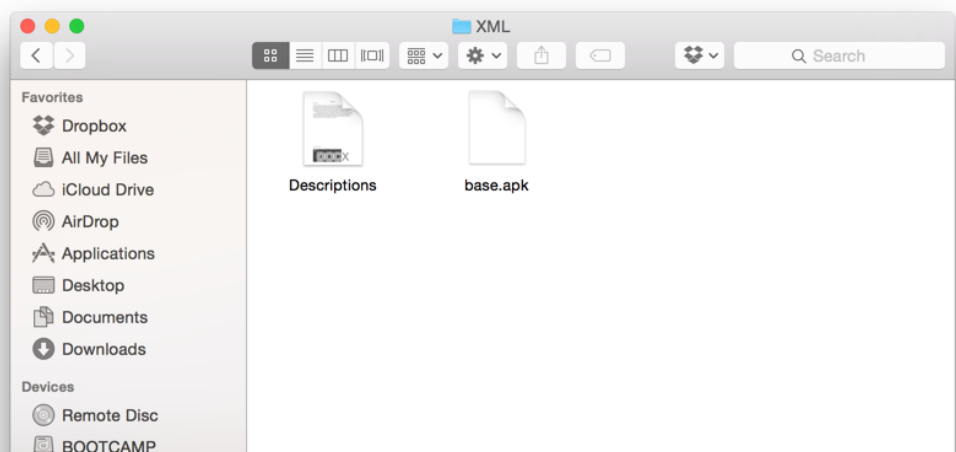




2. To pull the application package off the device:
 - i. Retrieve the APK from the device. Start by ensuring that you still have the virtual device running. Then, open Terminal or Command Prompt.
 - ii. Enter the following commands below:
 - The path to access adb (the Android debugger) varies among different platforms. The path below is where you will find adb on Mac OS X.
`cd ~/Library/Android/sdk/platform-tools`
 - On Windows, adb can be found in:
`C:\Users\YOUR_USERNAME_HERE\AppData\Local\Android\sdk\platform-tools`
 - iii. If you have forgotten the name of the package you want to work with, run
`./adb shell pm list packages`
 - iv. The following step is to get the full path, or where the package is located.
`./adb shell pm path YOUR_PACKAGE_NAME_GOES_HERE`
 - v. The output will look like:
`package:/data/app/com.example.someapp.apk`
We want the part that comes after “package:”.
 - vi. After we have the full path, pull the package off the device and onto the computer.
`./adb pull /data/app/com.example.someapp.apk
/PATH/TO/DESTINATION/GOES/HERE`



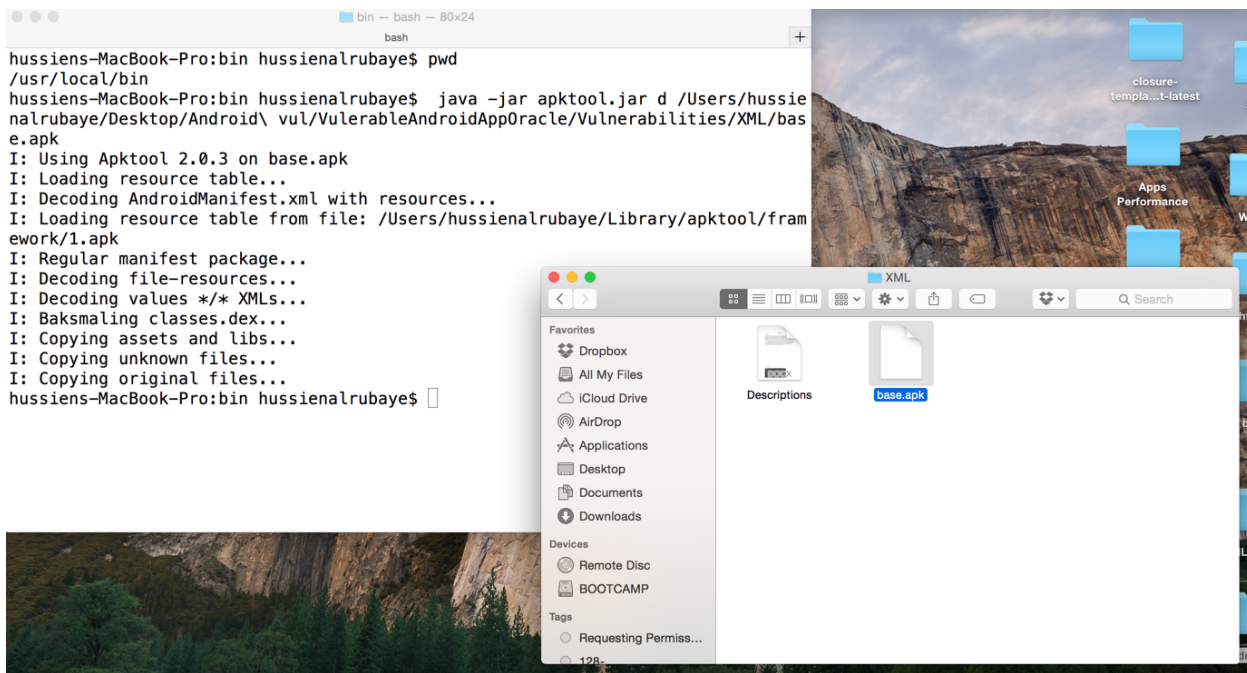
```
platform-tools — bash — 80x24
bash
hussiens-MacBook-Pro:platform-tools hussienalrubaye$ ./adb pull /data/app/com.example.hussienalrubaye.androidgooglemap-1/base.apk /Users/hussienalrubaye/Desktop/Android/vul/VulnerableAndroidAppOracle/Vulnerabilities/XML
2076 KB/s (3712654 bytes in 1.746s)
hussiens-MacBook-Pro:platform-tools hussienalrubaye$
```



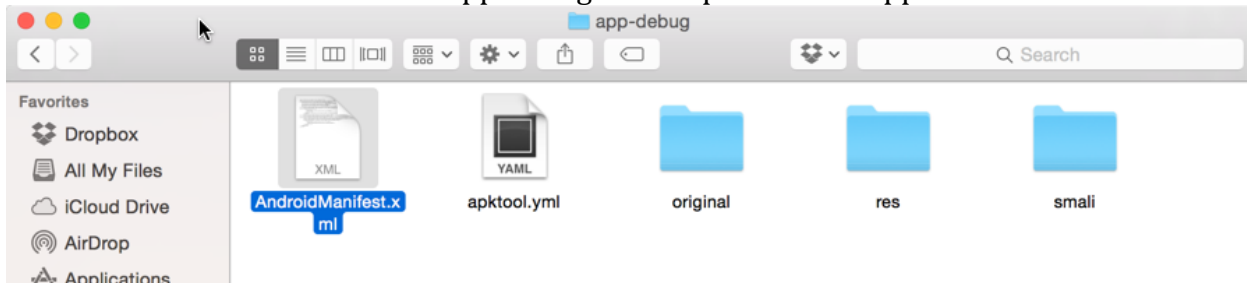
3. We have the app package, called “base.apk” here. The APK can be broken down to its source code in one of two ways:



- i. Rename the “base.apk” file to “base.zip”, and double click to open.
- ii. If method A did not work, try using a tool. Download **apktool** from here.
<http://ibotpeaches.github.io/Apktool/>
and install it as described on the website.
 - a. Run this command, from same path you installed **apktool.jar**
`java -jar apktool.jar d package_path`



4. You will see a folder named “app-debug” that represents all app files.



5. Open the file named “google_maps_api.xml”. In it, you can see the same API key that you created and edited in earlier.

This is an issue because your API key could be misused.



```
line:
76:A0:7B:52:24:16:38:F1:0F:B6:D3:CF:8F:6B:85:5F:
92:91:69:DC;com.example.hussienalrubaye.androidgooglemap

Alternatively, follow the directions here:
https://developers.google.com/maps/documentation/android/start#get-key

Once you have your key (it starts with "AIza"), replace the
"google_maps_key"YOUR_KEY_HERE
string in this file.
-->
<string name="google_maps_key" templateMergeStrategy="preserve"
translatable="false">AIzaSyDbCBLTfXoavK7GlyfqA1rDLS8JF_EbDAY</string>
</resources>
```

Defense

There are multiple ways to attack the problem. One solution is to only keep client keys on the device, retrieving API keys from a server you control via web requests. Essentially, we never store API keys in the APK because we assume that it can and will be decompiled.

Another is to use a tool to obfuscate your code, like ProGuard. In this approach, we would be trying to make it harder for people to reverse engineer the APK.