XML

Background:

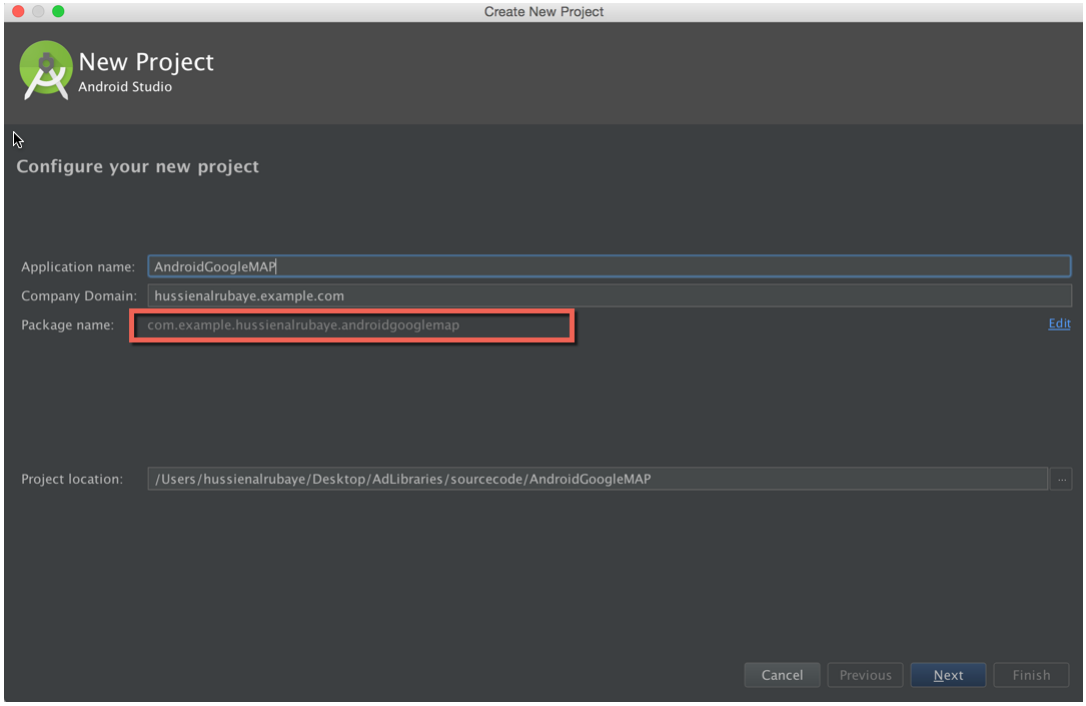
Android uses XML files to store static data, meaning that the data does not change through execution of the app. For example, translated strings (in languages such as English, Spanish, or Arabic) are stored in XML files for multi-language support. APK files are compressed files that contain many XML 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 charges on the developer's account.

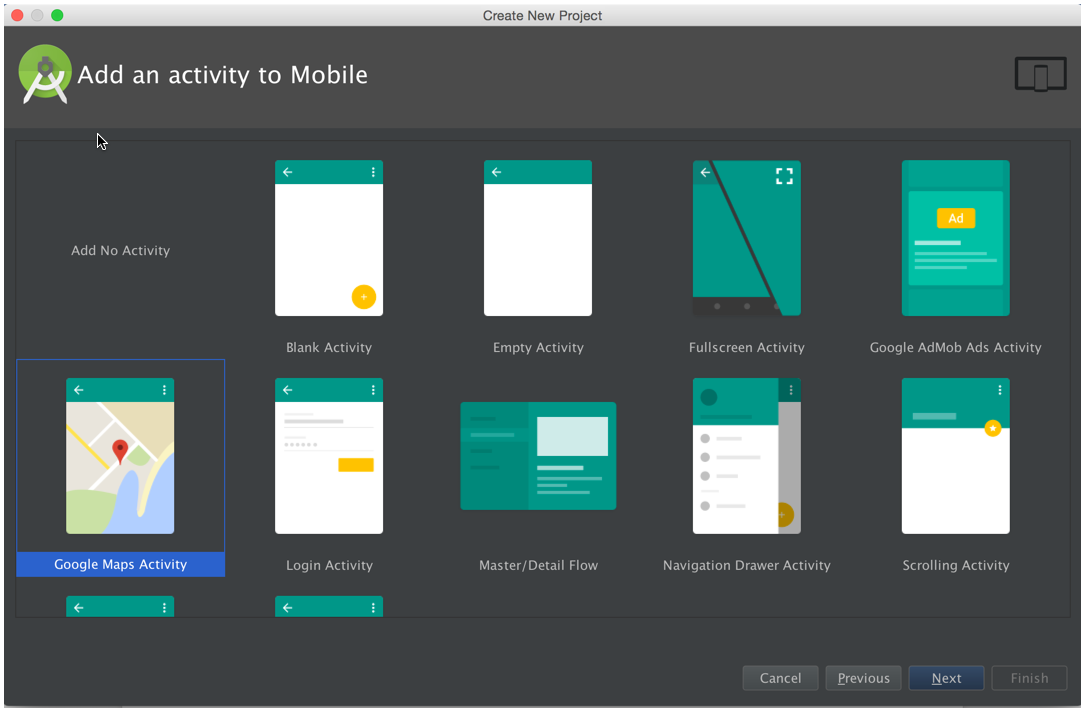
We will demonstrate how to build an Android app that displays Google Maps with the API key saved in the XML. After that, we will extract the key from the APK, and learn the best place to store the key.

Steps to Build the App:

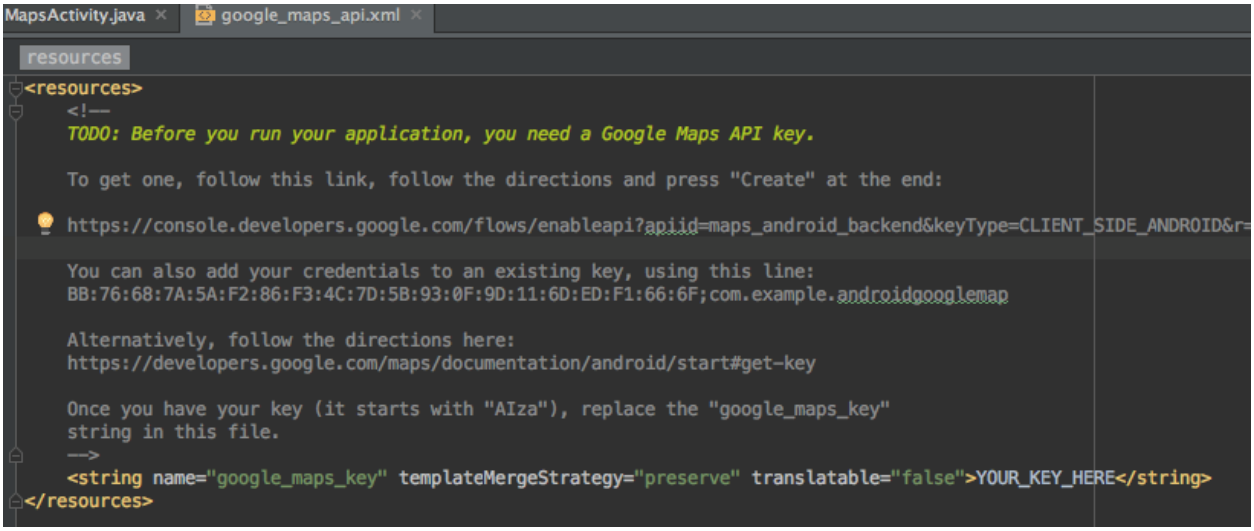
1. Create a new project named “AndroidGoogleMAP”. Make sure to remember package name, outlined in red below.



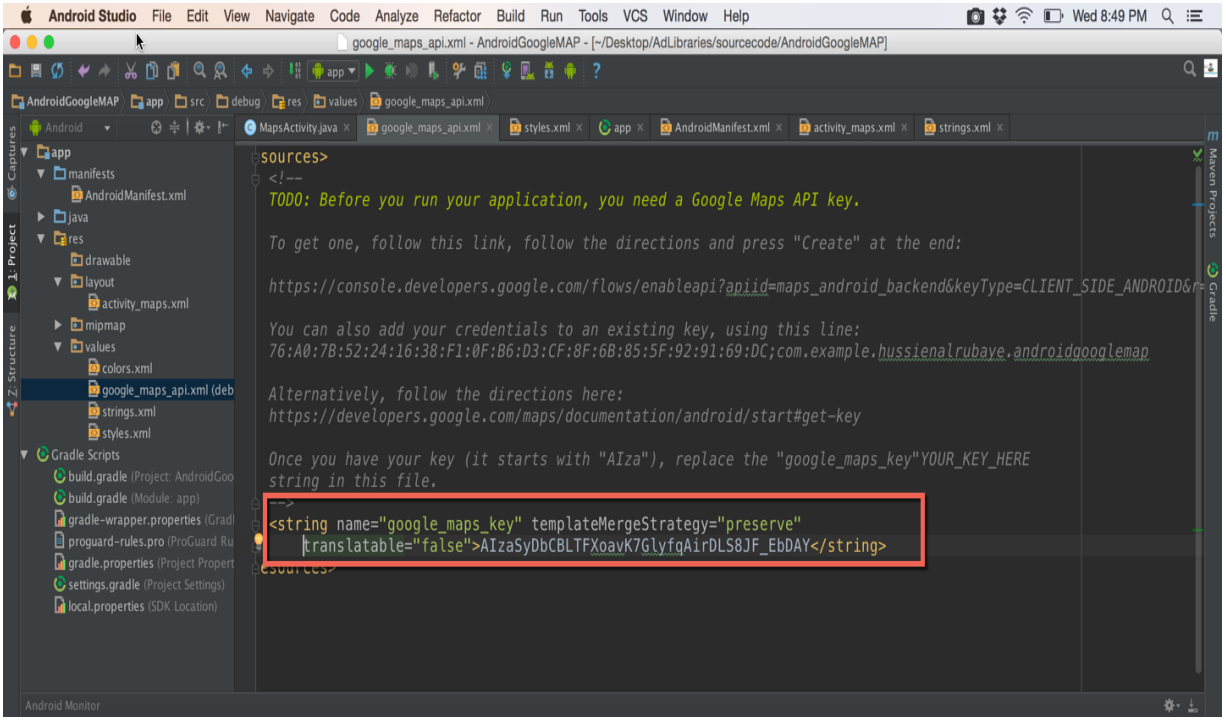
2. Select project of type **Google Maps Activity.**



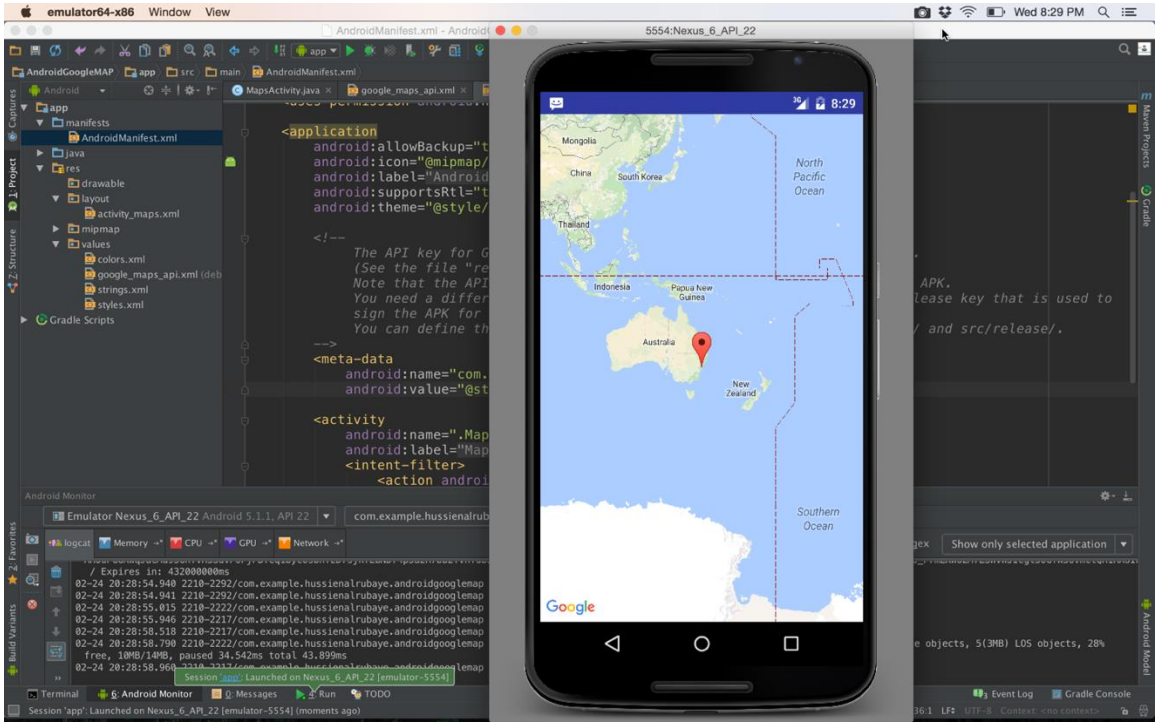
3. See the app files under “app/res/values/”. One of them is **strings.xml** file. It has a key named “google_maps_key”. We must create the Google Maps key for this app from the Google developer console by following the instructions in this file. After the key is created, replace the placeholder “YOUR_KEY_HERE”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



4. After replacing, you should see something like this:



5. Build and run the app.



How to Reverse Engineer an APK Back to Source Code:

1. Retrieve the APK from the device. Start by ensuring that you still have the virtual device running. Then, open Terminal or Command Prompt. 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
   ```

   If you have forgotten the name of the package you want to work with, run

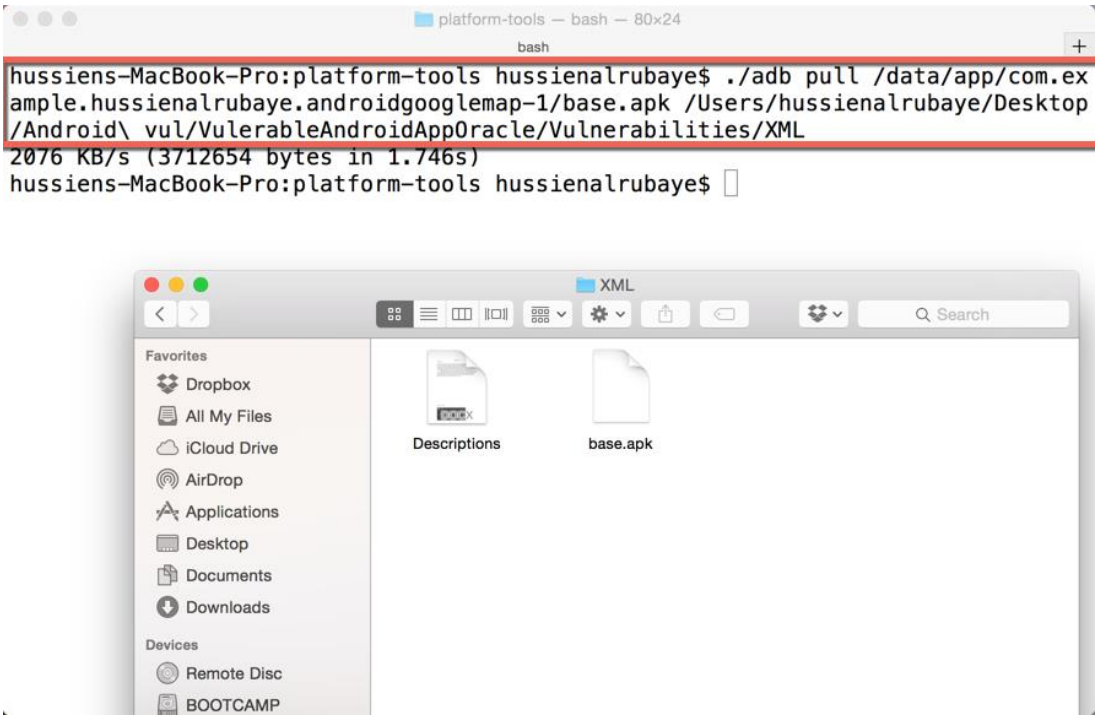
   ```
   ./adb shell pm list packages
   ```

   The following step is to get the full path, or where the package is located.

   ```
   ./adb shell pm path YOUR_PACKAGE_NAME_GOES_HERE
   ```

   The output will look like: package:/data/app/com.example.someapp.apk
   We want the part that comes after "package:".
   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
   ```



We have the app package, called “base.apk” here.

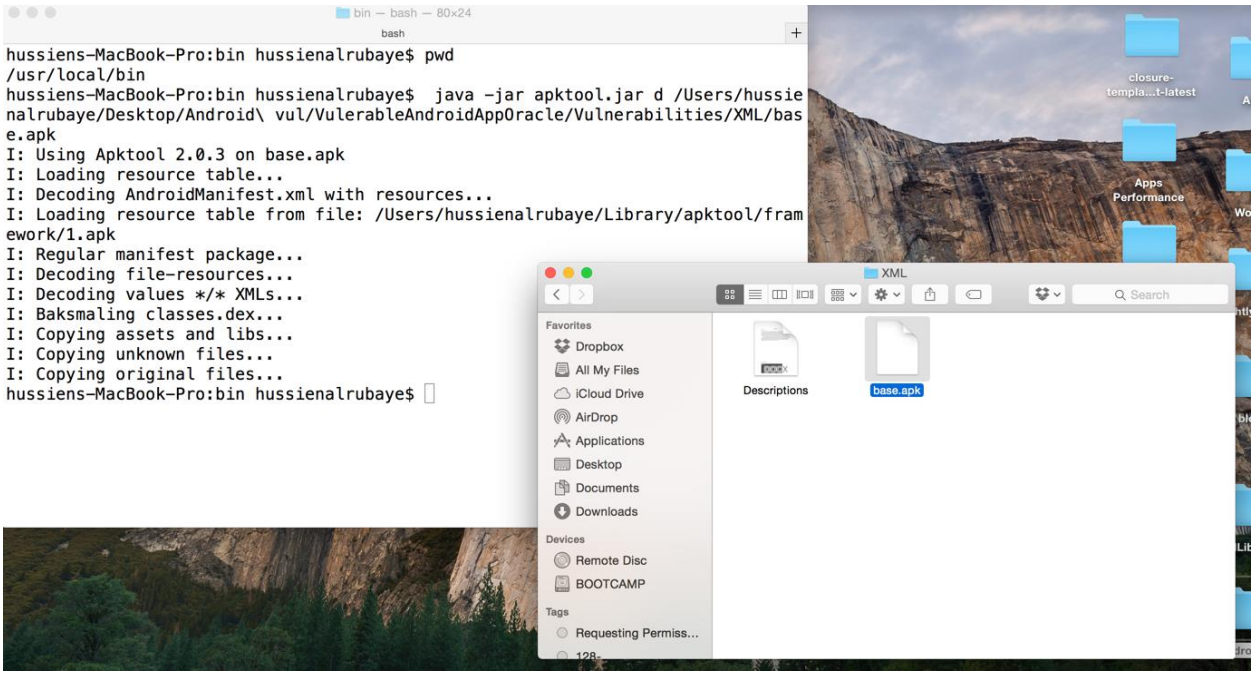
The APK can be broken down to its source code in one of two ways:

A. Rename the “base.apk” file to “base.zip”, and double click to open.
B. If method A did not work, try using a tool. Download **apktool** from here.
   http://ibotpeaches.github.io/Apktool/

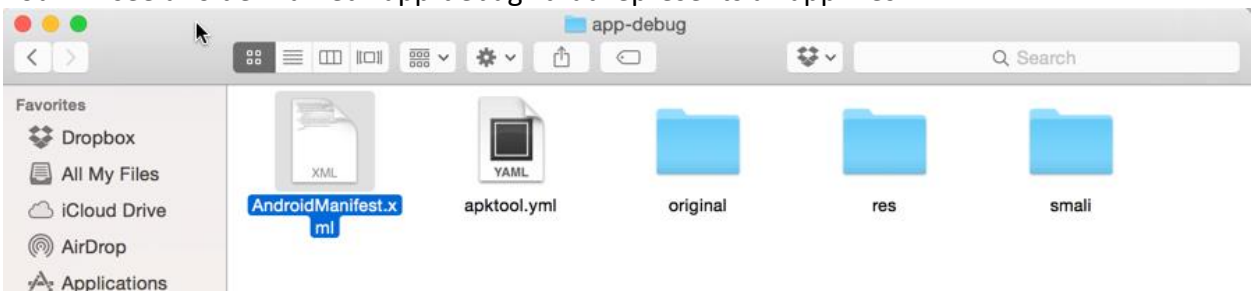
and install it as described on the website.

Run this command, from same path you installed **apktool.jar**

java –jar apktool.jar d package_path

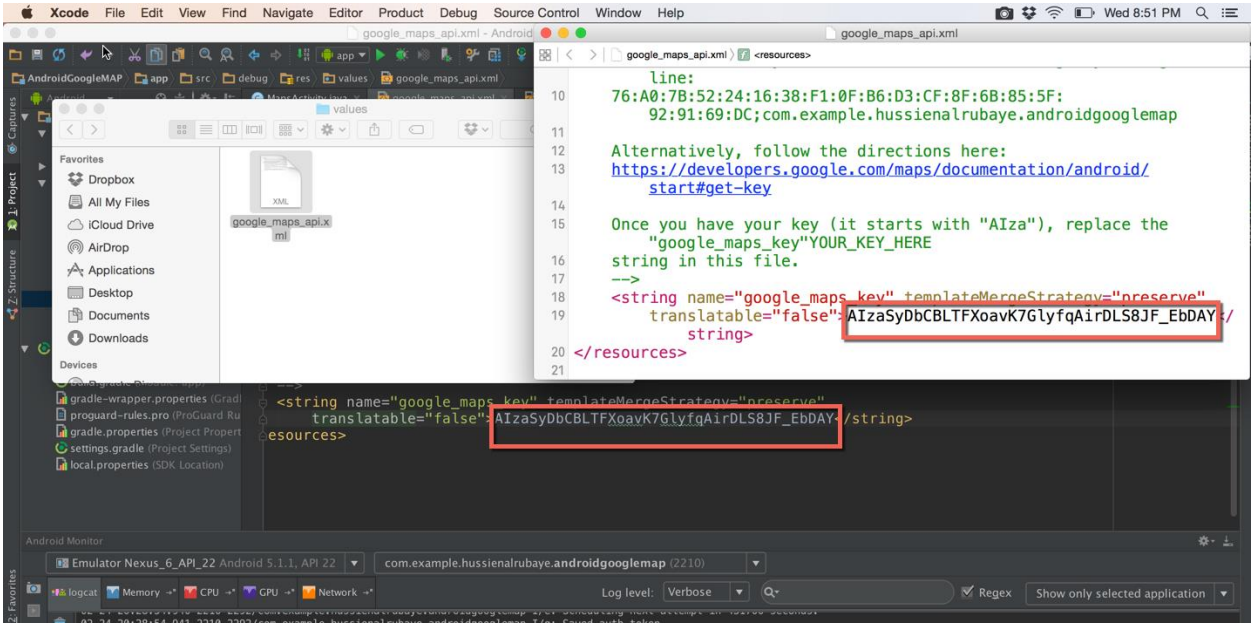


You will see a folder named “app-debug” that represents all app files.



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.



How to fix the problem:

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.