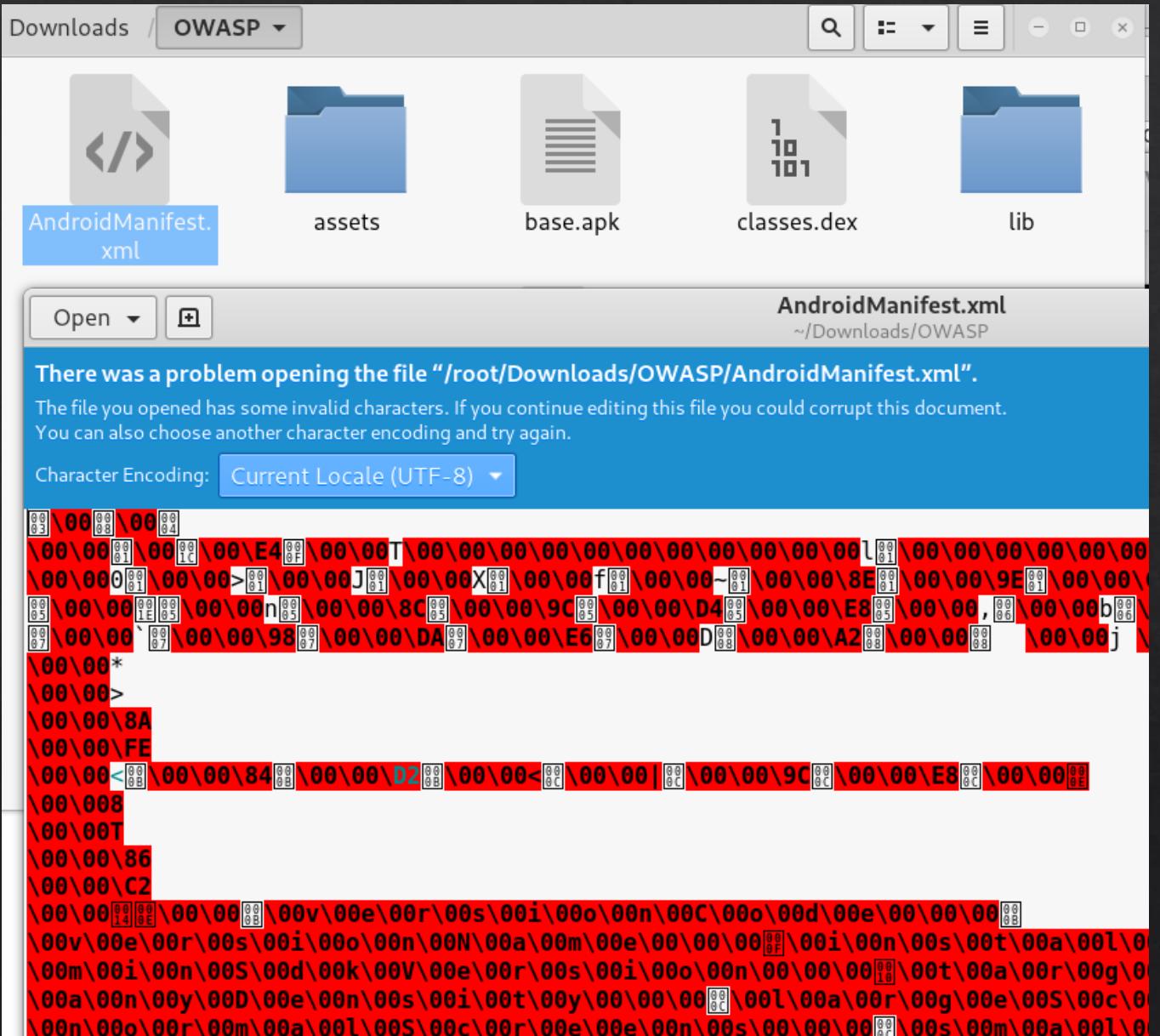


Let's open the  
AndroidManifest.xml file  
from unzipped APK →



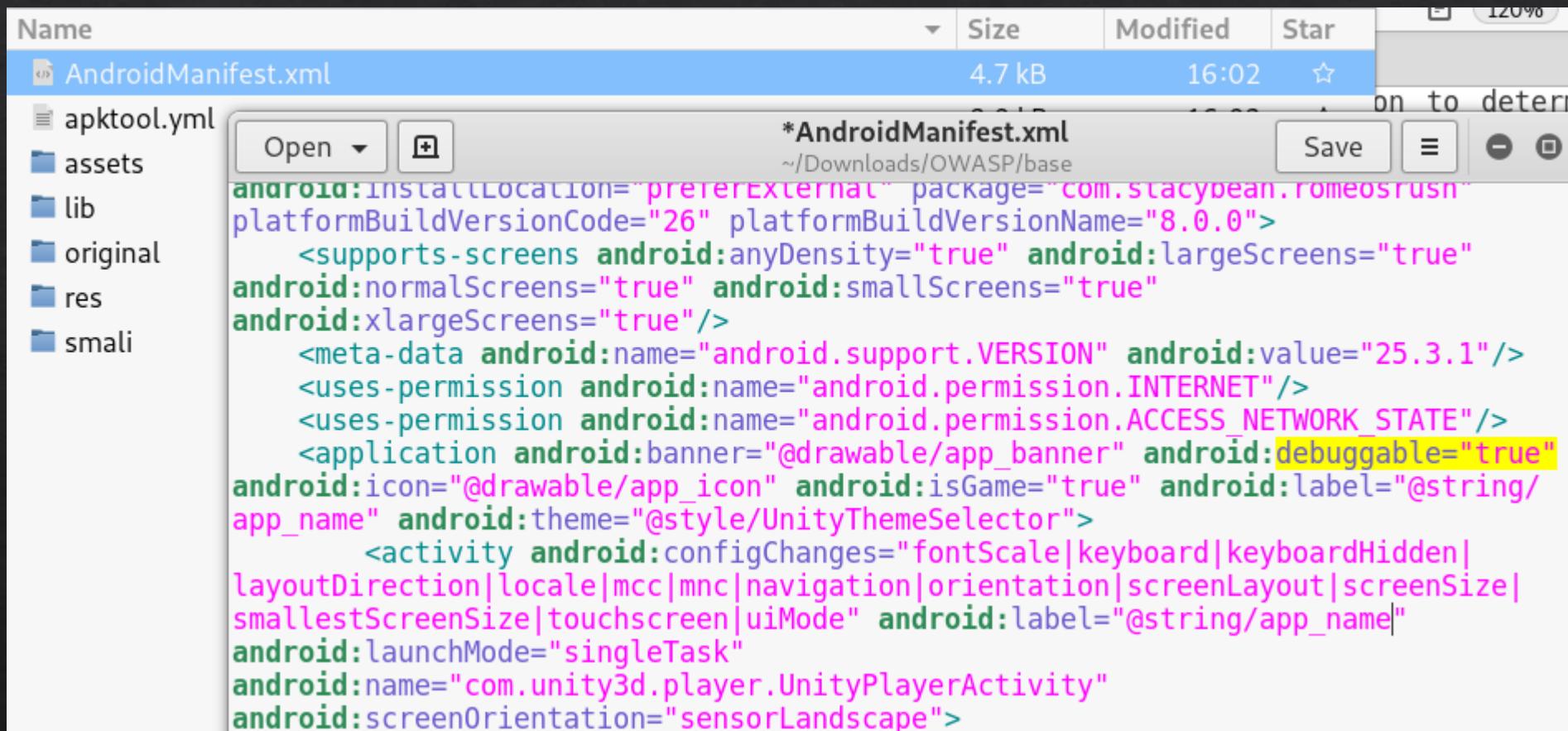
Okay so we need to decompile the APK properly so let's decompile it using APKTOOL →

```
root@Black-Box:~/Downloads/OWASP# apktool d base.apk
I: Using Apktool 2.3.4-dirty on base.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /root/.local/share/apktool/r
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
```

YAY ☺ let's see what's in there after decompiling →

```
root@Black-Box:~/Downloads/OWASP/base# ls -al
total 48
drwxr-xr-x  7 root root 4096 Feb 21 16:02 .
drwxr-xr-x  7 root root 4096 Feb 21 16:02 ..
-rw-r--r--  1 root root 4722 Feb 21 16:02 AndroidManifest.xml
-rw-r--r--  1 root root 8864 Feb 21 16:02 apktool.yml
drwxr-xr-x  3 root root 4096 Feb 21 16:02 assets
drwxr-xr-x  4 root root 4096 Feb 21 16:02 lib
drwxr-xr-x  3 root root 4096 Feb 21 16:02 original
drwxr-xr-x 162 root root 4096 Feb 21 16:02 res
drwxr-xr-x  7 root root 4096 Feb 21 16:02 smali
```

Now let's modify the `AndroidManifest.xml` file and set the debuggable option to True so we can later hook this app into a debugger 😊



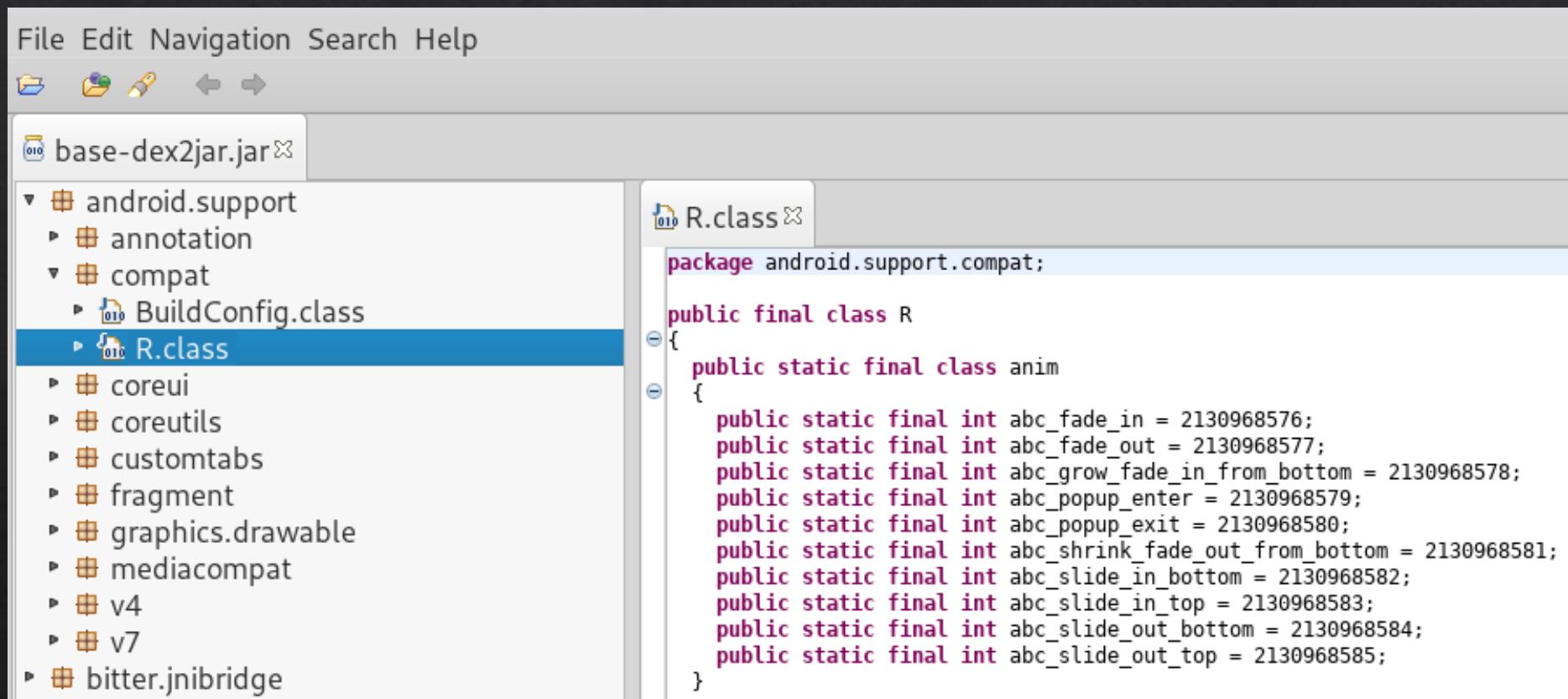
The screenshot shows a file manager interface with a sidebar containing files like apktool.yml, assets, lib, original, res, and smali. The main area displays the contents of the `AndroidManifest.xml` file. The file path is shown as `~/Downloads/OWASP/base`. The XML code includes several `uses-permission` tags and an `application` tag. Within the `application` tag, the `android:debuggable` attribute is set to `true`, which is highlighted with a yellow background.

```
        android:installLocation="preferExternal" package="com.stacybean.romeosrush"
        platformBuildVersionCode="26" platformBuildVersionName="8.0.0">
            <supports-screens android:anyDensity="true" android:largeScreens="true"
        android:normalScreens="true" android:smallScreens="true"
        android:xlargeScreens="true"/>
            <meta-data android:name="android.support.VERSION" android:value="25.3.1"/>
            <uses-permission android:name="android.permission.INTERNET"/>
            <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
            <application android:banner="@drawable/app_banner" android:debuggable="true"
        android:icon="@drawable/app_icon" android:isGame="true" android:label="@string/
        app_name" android:theme="@style/UnityThemeSelector">
                <activity android:configChanges="fontScale|keyboard|keyboardHidden|
        layoutDirection|locale|mcc|mnc|navigation|orientation|screenLayout|screenSize|
        smallestScreenSize|touchscreen|uiMode" android:label="@string/app_name|
        android:launchMode="singleTask"
                android:name="com.unity3d.player.UnityPlayerActivity"
                android:screenOrientation="sensorLandscape">
```

You can also convert an APK file directly into a JAR format using dex2jar →

```
root@Black-Box:~/Downloads/OWASP# d2j-dex2jar base.apk
dex2jar base.apk -> base-dex2jar.jar
root@Black-Box:~/Downloads/OWASP# ls
AndroidManifest.xml assets base base.apk base-dex2jar.jar
root@Black-Box:~/Downloads/OWASP# |
```

Once done, we can then open this .jar file in JD – GUI →



We are now ready to recompile our app back to .APK file. We will use APKTOOL again to achieve this.

```
root@Black-Box:~/Downloads/0WASP# apktool b base
I: Using Apktool 2.3.4-dirty
I: Checking whether sources has changed...
I: Checking whether resources has changed...
I: Building apk file...
I: Copying unknown files/dir...
I: Built apk...
root@Black-Box:~/Downloads/0WASP# cd base/dist
root@Black-Box:~/Downloads/0WASP/base/dist# ls -al
total 78844
drwxr-xr-x 2 root root      4096 Feb 21 16:21 .
drwxr-xr-x 9 root root      4096 Feb 21 16:21 ..
-rw-r--r-- 1 root root 80727908 Feb 21 16:21 base.apk
root@Black-Box:~/Downloads/0WASP/base/dist#
```

And now it is time to sign the newly compiled APK file with the Android test certificate. I am using APK Sign for this. <https://github.com/appium/sign>

```
root@Black-Box:~/Downloads/sign# ls
build.sh dist license pom.xml readme.md signapk_jar sign.bat sign_jar src testkey.pk8 testkey.x509.pem
root@Black-Box:~/Downloads/sign# java -jar dist/signapk.jar testkey.x509.pem testkey.pk8 ~/Downloads/OWASP/base/dist/base.apk signed.apk
root@Black-Box:~/Downloads/sign# ls -al
total 78856
drwxr-xr-x  8 root root    4096 Feb 21 16:28 .
drwxr-xr-x 16 root root    4096 Feb 21 15:30 ..
-rwxr-xr-x  1 root root     285 Feb 18 22:59 build.sh
drwxr-xr-x  2 root root    4096 Feb 20 20:14 dist
drwxr-xr-x  8 root root    4096 Feb 18 22:59 .git
-rw-r--r--  1 root root      48 Feb 18 22:59 .gitignore
drwxr-xr-x  3 root root    4096 Feb 18 22:59 license
-rw-r--r--  1 root root   1784 Feb 18 22:59 pom.xml
-rw-r--r--  1 root root   2130 Feb 18 22:59 readme.md
drwxr-xr-x  3 root root    4096 Feb 18 22:59 signapk_jar
-rw-r--r--  1 root root    176 Feb 18 22:59 sign.bat
-rw-r--r--  1 root root 80679901 Feb 21 16:29 signed.apk
drwxr-xr-x  3 root root    4096 Feb 18 22:59 sign_jar
drwxr-xr-x  4 root root    4096 Feb 18 22:59 src
-rw-r--r--  1 root root   1217 Feb 18 22:59 testkey.pk8
-rw-r--r--  1 root root   1675 Feb 18 22:59 testkey.x509.pem
```

We will use Zipalign (part of Android SDK build-tools) to optimise our newly signed APK. You can read more about Zipalign here:

<https://developer.android.com/studio/command-line/zipalign>

```
root@Black-Box:~/Downloads/sign# zipalign -v 4 signed.apk base.apk
Verifying alignment of base.apk (4)...
    53 AndroidManifest.xml (OK - compressed)
   2329 assets/bin/Data/Managed/Assembly-CSharp-firstpass.dll (OK - compressed)
   9671 assets/bin/Data/Managed/Assembly-CSharp.dll (OK - compressed)
  179442 assets/bin/Data/Managed/Facebook.Unity.Android.dll (OK - compressed)
  180936 assets/bin/Data/Managed/Facebook.Unity.Gameroom.dll (OK - compressed)
  183847 assets/bin/Data/Managed/Facebook.Unity.iOS.dll (OK - compressed)
  184951 assets/bin/Data/Managed/Facebook.Unity.Settings.dll (OK - compressed)
  188108 assets/bin/Data/Managed/Facebook.Unity.dll (OK - compressed)
  226598 assets/bin/Data/Managed/FacebookNamedPipeClient.dll (OK - compressed)
  237656 assets/bin/Data/Managed/Mono.Security.dll (OK - compressed)
  354630 assets/bin/Data/Managed/System.Core.dll (OK - compressed)
  455878 assets/bin/Data/Managed/System.Xml.dll (OK - compressed)
```

Let's push this to mobile  
now →

```
root@Black-Box:~/Downloads/sign# adb install base.apk
Success
root@Black-Box:~/Downloads/sign# |
```

And we successfully managed to change the name of the app and installed it on the phone ☺



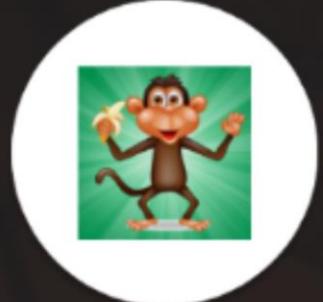
Keep Not...



LinkedIn



OlaCabs



OWASP



- Use code obfuscator like ProGuard which is a cross platform tool written in Java.

Original Source Code Before Rename Obfuscation	Reverse-Engineered Source Code After Rename Obfuscation
<pre>private void CalculatePayroll(SpecialList employeeGroup) {     while(employeeGroup.HasMore()) {         employee = employeeGroup.GetNext(true);         employee.UpdateSalary();         DistributeCheck(employee);     } }</pre>	<pre>private void a(a b) {     while (b.a()) {         a = b.a(true);         a.a();         a(a);     } }</pre>

- Developers can also obfuscate code manually. Code obfuscation may include:
  - encrypting some or all of the code;
  - stripping out potentially revealing metadata
  - renaming useful class and variable names to meaningless labels;
  - adding unused or meaningless code to an application's binary.

- **Do Not store any sensitive information in the application source code**
- **Move important code chunks to the server side**
- **Do Not store user credentials at the device level and rather store a short lived encrypted token**
- **Will highly recommend not to store API keys in the code and make use of public/private key exchange to protect your API**
- **Do Not store app data to external storage since it can be read by all of the applications.**
- **Enable Anti-Tamper and Anti-Debug by injecting a self-protection code into the application source code. E.g. Random crashes and limited functionality upon tampering detection.**

## Further reading:

OWASP Mobile Top 10

[https://www.owasp.org/index.php/Mobile Top 10 2016-Top 10](https://www.owasp.org/index.php/Mobile_Top_10_2016-Top_10)

<https://hackernoon.com/mobile-application-security-best-practices-for-app-developers-1a6345750b35>

<https://www.multidots.com/how-to-prevent-reverse-engineering-of-your-mobile-apps/>

<https://www.preemptive.com/obfuscation>

<https://www.talentica.com/blogs/reverse-engineering-of-mobile-apps/>

A large, stylized word cloud centered around the words "Thank You" in various languages. The word "Thank" is at the top left, "You" is at the bottom center, and "Merci" is to its left. The words are in different sizes and colors (red, orange, yellow, green, blue) and are arranged in a radial pattern from the center outwards. Each word is accompanied by one or more smaller words in the same language, likely representing dialects or related terms.