Reverse Engineering

KeepassDroid

Yewale Pankaj Subhash

psy231@nyu.edu

Course : Application Security (CS-GY 9163)

Faculty : Professor Kelly Lum

Semester : Fall 2016

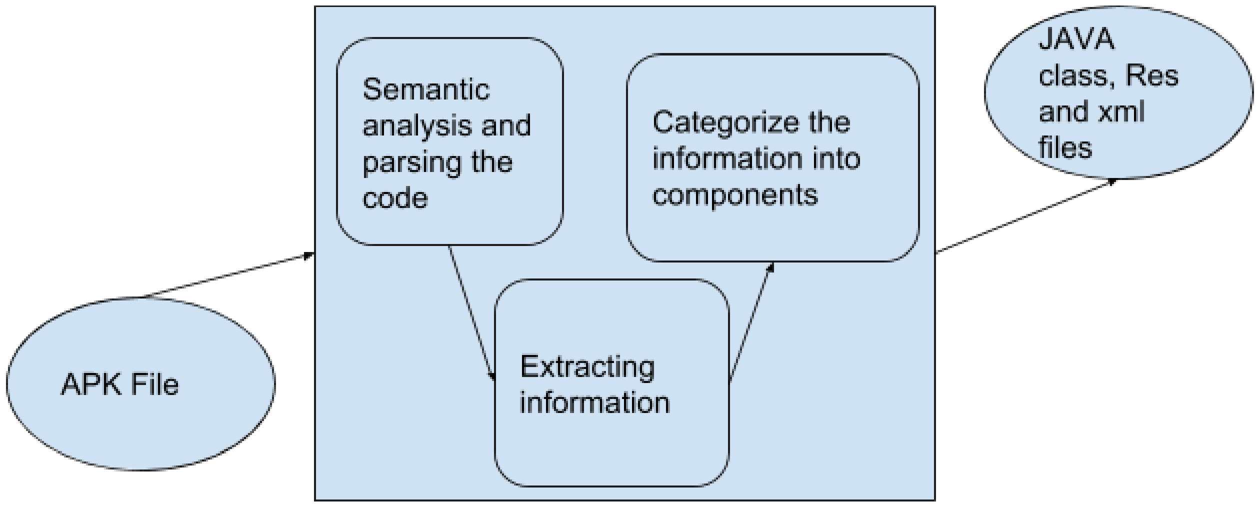
Introduction

Reverse Engineering requires highlighting of technical aspects in the system or application. This requires thorough analysis of its architecture, working and functioning to get to know more about its behaviour under worst and best case scenarios. Applications are compiled to work in byte code mode. Reverse engineering needs to decompile that byte code and extract the source code of the application. It's a way to understand the engineering and functioning behind any software and build a new software on top of it to reverse its functioning.

In this assessment, reverse engineering for an Android application will end up in extracting the source Java code from a binary file which is in the form of APK. Obfuscation is a constructive hurdle in the process of reverse engineering. It cannot disable reverse engineering process but makes things look complex and increases the time to crack the apk.

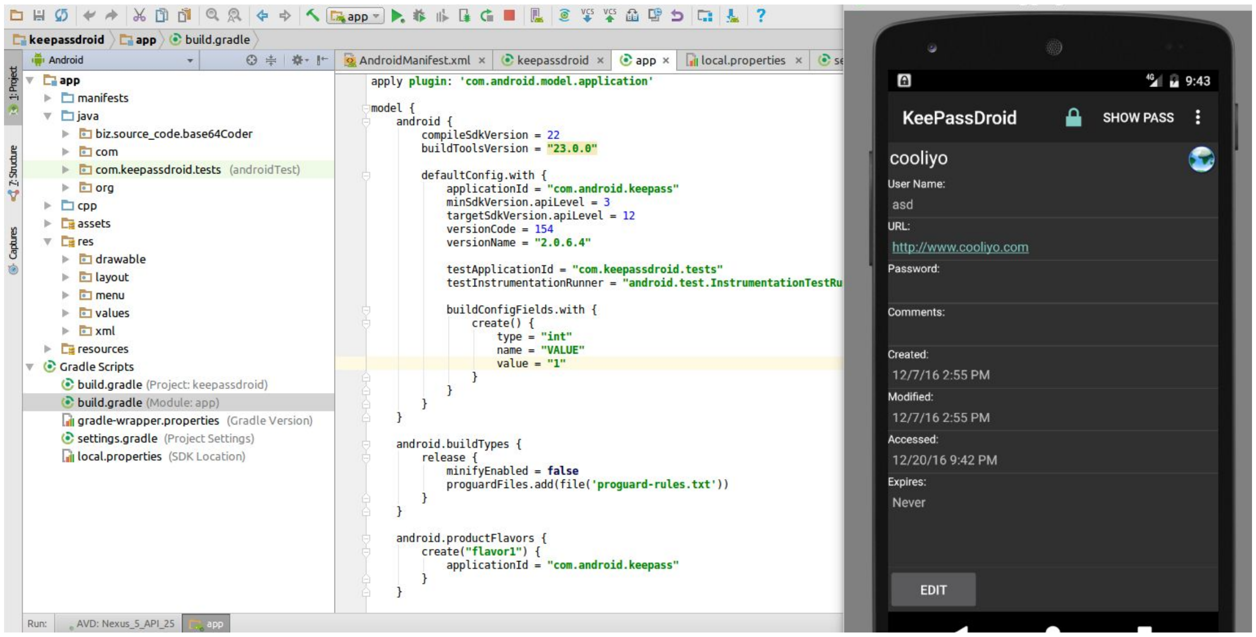
Reverse engineering deploys three core processes.

- Parsing and semantic analysis of code
- Extracting information from the code
- Dividing the product into components


APK File
Semantic analysis and parsing the code
Extracting information
Categorize the information into components
JAVA class, Res and xml files

Keepassdroid

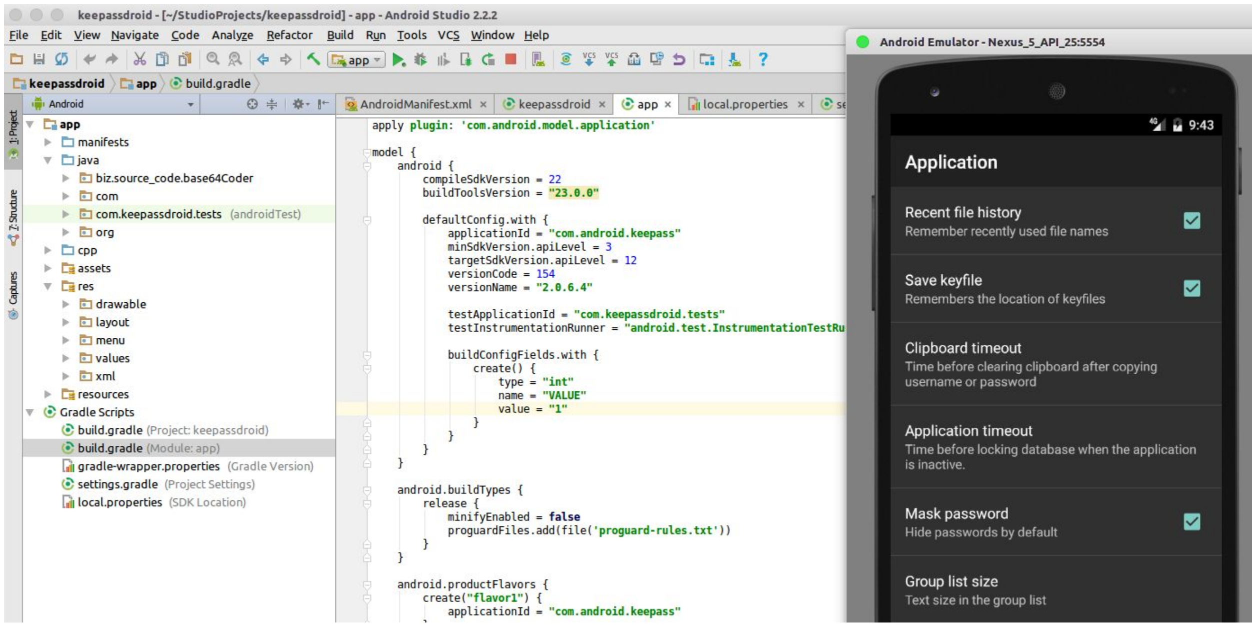
Keepassdroid[4] is a password manager for android devices. It manages your passwords in a secured way in one DB locked with a key. Then you can store all your passwords of different websites and use this key to unlock the DB. Secure encryption algorithms like AES are used. Following are the screenshots from my android studio by cloning the Keepassdroid repo.



```
app
  manifests
  java
    biz.source_code.base64Coder
    com
    com.keepassdroid.tests (androidTest)
    org
  cpp
  assets
  res
    drawable
    layout
    menu
    values
    xml
  resources
Gradle Scripts
  build.gradle (Project: keepassdroid)
  build.gradle (Module: app)
  gradle-wrapper.properties (Gradle Version)
  settings.gradle (Project Settings)
  local.properties (SDK Location)
```

```
apply plugin: 'com.android.model.application'

model {
    android {
        compileSdkVersion = 22
        buildToolsVersion = "23.0.0"

        defaultConfig.with {
            applicationId = "com.android.keepass"
            minSdkVersion.apiLevel = 3
            targetSdkVersion.apiLevel = 12
            versionCode = 154
            versionName = "2.0.6.4"

            testApplicationId = "com.keepassdroid.tests"
            testInstrumentationRunner = "android.test.InstrumentationTestRu

            buildConfigFields.with {
                create() {
                    type = "int"
                    name = "VALUE"
                    value = "1"
                }
            }
        }
    }

    android.buildTypes {
        release {
            minifyEnabled = false
            proguardFiles.add(file('proguard-rules.txt'))
        }
    }

    android.productFlavors {
        create("flavor1") {
            applicationId = "com.android.keepass"
        }
    }
```

9:43
KeePassDroid SHOW PASS
cooliyo
User Name: asd
URL: http://www.cooliyo.com
Password:
Comments:
Created: 12/7/16 2:55 PM
Modified: 12/7/16 2:55 PM
Accessed: 12/20/16 9:42 PM
Expires: Never
EDIT


keepassdroid - [~/StudioProjects/keepassdroid] - app - Android Studio 2.2.2
File
Edit
View
Navigate
Code
Analyze
Refactor
Build
Run
Tools
VCS
Window
Help
Android Emulator - Nexus_5_API_25:5554

```
apply plugin: 'com.android.model.application'

model {
    android {
        compileSdkVersion = 22
        buildToolsVersion = "23.0.0"

        defaultConfig.with {
            applicationId = "com.android.keepass"
            minSdkVersion.apiLevel = 3
            targetSdkVersion.apiLevel = 12
            versionCode = 154
            versionName = "2.0.6.4"

            testApplicationId = "com.keepassdroid.tests"
            testInstrumentationRunner = "android.test.InstrumentationTestRu

            buildConfigFields.with {
                create() {
                    type = "int"
                    name = "VALUE"
                    value = "1"
                }
            }
        }
    }

    android.buildTypes {
        release {
            minifyEnabled = false
            proguardFiles.add(file('proguard-rules.txt'))
        }
    }

    android.productFlavors {
        create("flavor1") {
            applicationId = "com.android.keepass"
```

9:43
Application
Recent file history — Remember recently used file names ✓
Save keyfile — Remembers the location of keyfiles ✓
Clipboard timeout — Time before clearing clipboard after copying username or password
Application timeout — Time before locking database when the application is inactive.
Mask password — Hide passwords by default ✓
Group list size — Text size in the group list

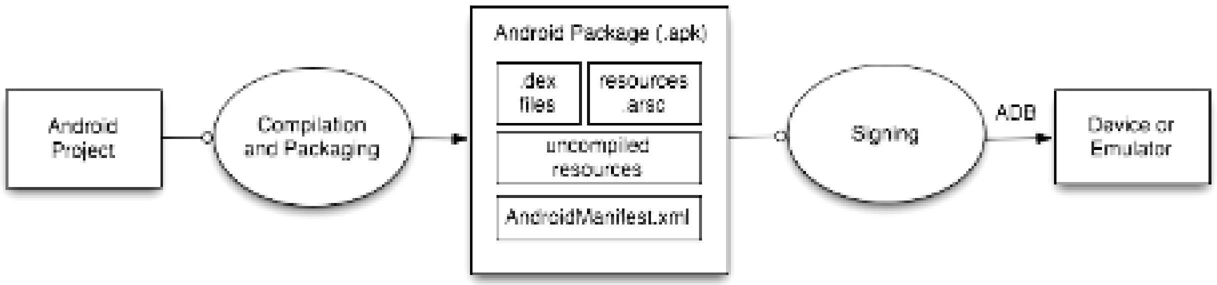
APK

Any application that you develop for or get from Google Play Store is formatted as APK(Android Application Package) [2] bundle. In short Apk bundle is the format used to package any application for Google Play Store. So, there exists an APK file for every application present on android devices. The native applications that come with factory defaults also have APKS. Android Application Package is similar to a ZIP file. So one quick hack is to rename it and then extract it. You get access to its content.

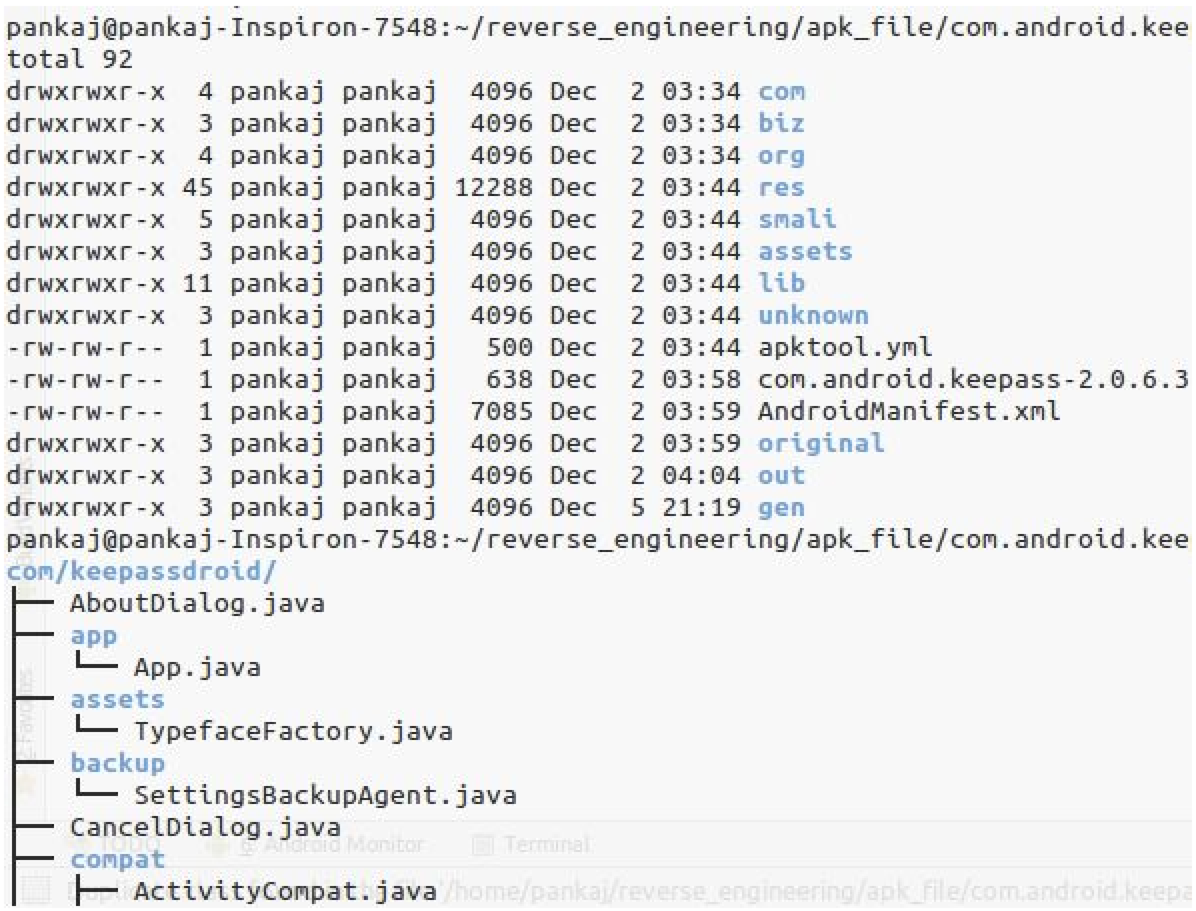
File/Folder	Description
AndroidManifest.xml	the manifest file in Binary XML format manifest file
resources.arsc	A binary XML file containing precompiled application resources
META-INF/	signature of the APK is stored in this file, folder containing the MANIFEST.MF file, metadata about the contents of the JAR
classes.dex	dex format compiled application code
res/	folder containing resources. These resources are not compiled into resources.arsc
assets/	Folder containing applications assets(optional) It can be accessed by AssetManager.
lib/	native code libraries(optional). Contained compiled code

Android application build process

Process from project to APK.



APK file content example from the reverse engineered apk.



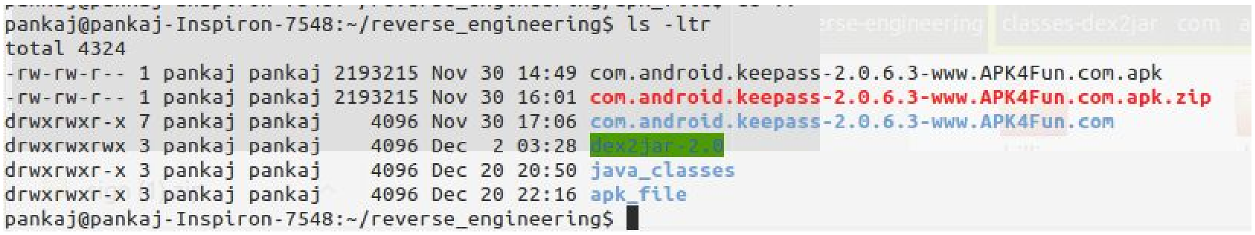
```
pankaj@pankaj-Inspiron-7548:~/reverse_engineering/apk_file/com.android.kee
total 92
drwxrwxr-x  4 pankaj pankaj  4096 Dec  2 03:34 com
drwxrwxr-x  3 pankaj pankaj  4096 Dec  2 03:34 biz
drwxrwxr-x  4 pankaj pankaj  4096 Dec  2 03:34 org
drwxrwxr-x 45 pankaj pankaj 12288 Dec  2 03:44 res
drwxrwxr-x  5 pankaj pankaj  4096 Dec  2 03:44 smali
drwxrwxr-x  3 pankaj pankaj  4096 Dec  2 03:44 assets
drwxrwxr-x 11 pankaj pankaj  4096 Dec  2 03:44 lib
drwxrwxr-x  3 pankaj pankaj  4096 Dec  2 03:44 unknown
-rw-rw-r--  1 pankaj pankaj   500 Dec  2 03:44 apktool.yml
-rw-rw-r--  1 pankaj pankaj   638 Dec  2 03:58 com.android.keepass-2.0.6.3
-rw-rw-r--  1 pankaj pankaj  7085 Dec  2 03:59 AndroidManifest.xml
drwxrwxr-x  3 pankaj pankaj  4096 Dec  2 03:59 original
drwxrwxr-x  3 pankaj pankaj  4096 Dec  2 04:04 out
drwxrwxr-x  3 pankaj pankaj  4096 Dec  5 21:19 gen
pankaj@pankaj-Inspiron-7548:~/reverse_engineering/apk_file/com.android.kee
com/keepassdroid/
├── AboutDialog.java
├── app
│   └── App.java
├── assets
│   └── TypefaceFactory.java
├── backup
│   └── SettingsBackupAgent.java
├── CancelDialog.java
├── compat
│   ├── ActivityCompat.java
```

Deployment

Step 1:

1. Copy the .apk file which is to be reverse engineered
2. Name the extension of .apk file to .zip

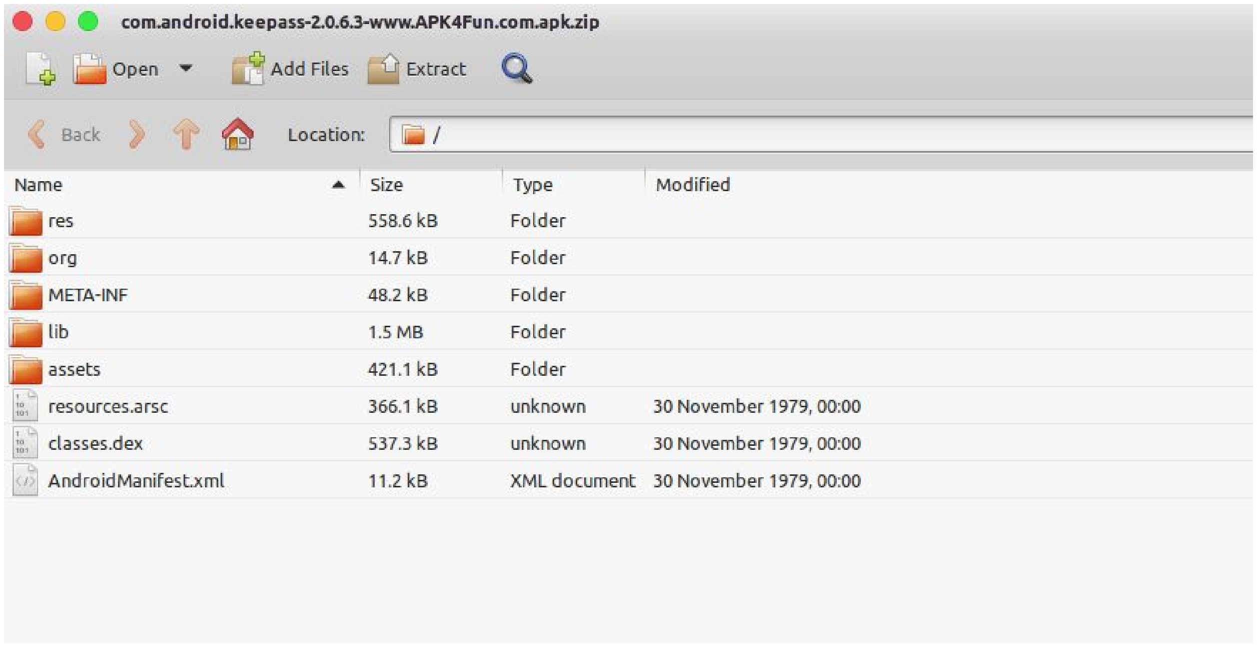
```
pankaj@pankaj-Inspiron-7548:~/reverse_engineering$ ls -ltr
total 4324
-rw-rw-r--   1   pankaj   pankaj   2193215   Nov   30   14:49
com.android.keepass-2.0.6.3-www.APK4Fun.com.apk
-rw-rw-r--   1   pankaj   pankaj   2193215   Nov   30   16:01
com.android.keepass-2.0.6.3-www.APK4Fun.com.apk.zip
drwxrwxr-x   7   pankaj   pankaj          4096   Nov   30   17:06
com.android.keepass-2.0.6.3-www.APK4Fun.com
drwxrwxrwx 3 pankaj pankaj    4096 Dec  2 03:28 dex2jar-2.0
drwxrwxr-x 3 pankaj pankaj    4096 Dec 20 20:50 java_classes
drwxrwxr-x 3 pankaj pankaj    4096 Dec 20 22:16 apk_file
pankaj@pankaj-Inspiron-7548:~/reverse_engineering$
```



```
pankaj@pankaj-Inspiron-7548:~/reverse_engineering$ ls -ltr
total 4324
-rw-rw-r-- 1 pankaj pankaj 2193215 Nov 30 14:49 com.android.keepass-2.0.6.3-www.APK4Fun.com.apk
-rw-rw-r-- 1 pankaj pankaj 2193215 Nov 30 16:01 com.android.keepass-2.0.6.3-www.APK4Fun.com.apk.zip
drwxrwxr-x 7 pankaj pankaj    4096 Nov 30 17:06 com.android.keepass-2.0.6.3-www.APK4Fun.com
drwxrwxrwx 3 pankaj pankaj    4096 Dec  2 03:28 dex2jar-2.0
drwxrwxr-x 3 pankaj pankaj    4096 Dec 20 20:50 java_classes
drwxrwxr-x 3 pankaj pankaj    4096 Dec 20 22:16 apk_file
pankaj@pankaj-Inspiron-7548:~/reverse_engineering$
```

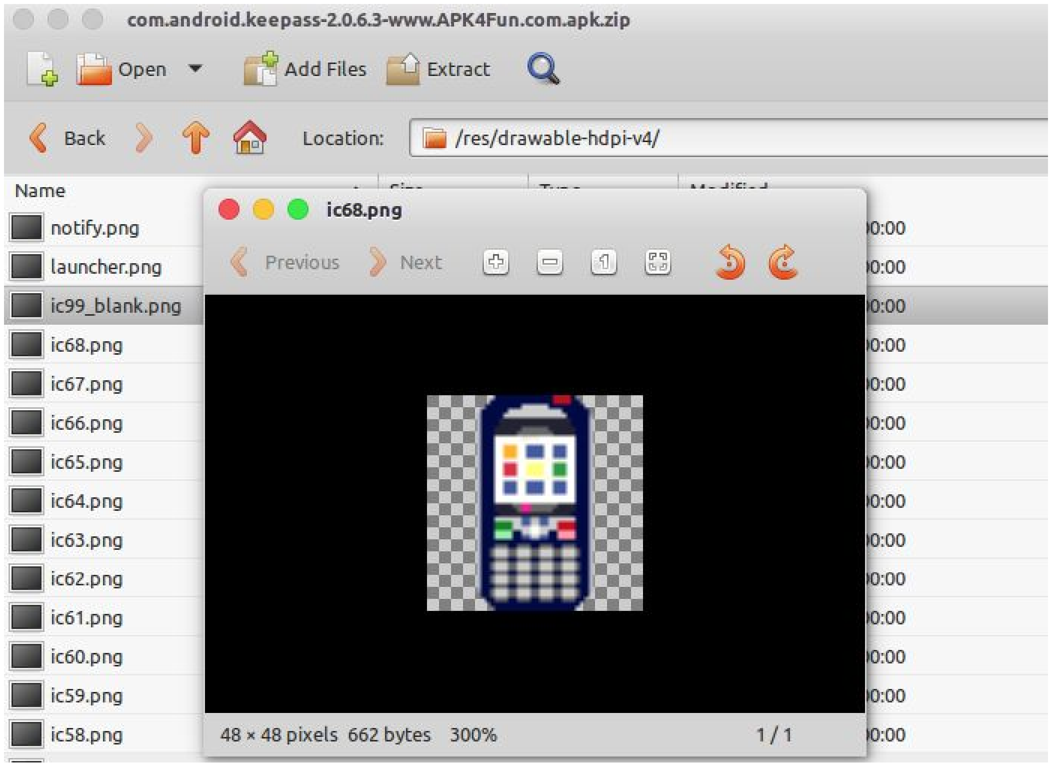
3. Classes.dex files can now be accessed.

   A compress file consisting of all java classes in the application package is a dex file. A jar file on other hand is a package of .class files. These .class files are isolated which makes jar and dex files differ from each other.

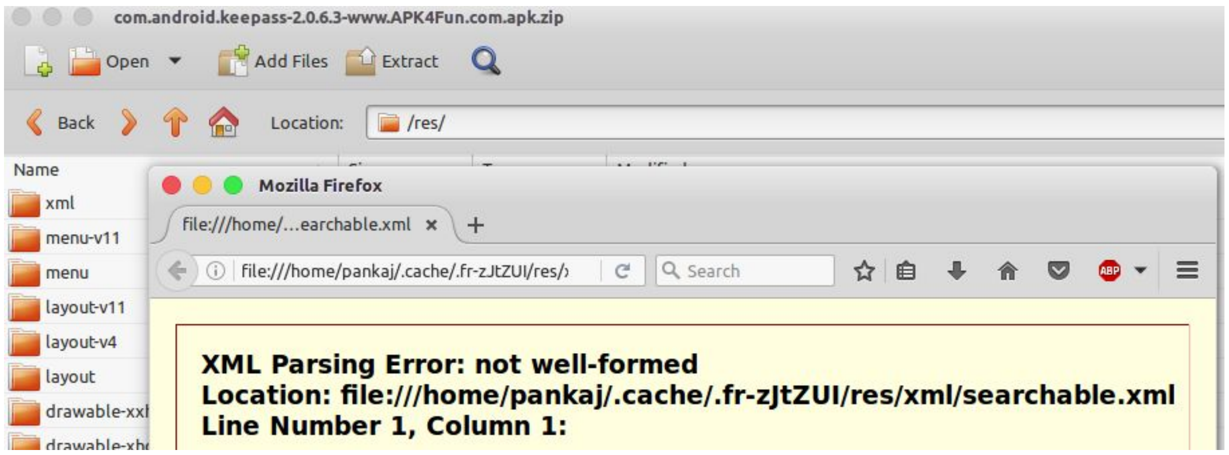


Name	Size	Type	Modified
res	558.6 kB	Folder	
org	14.7 kB	Folder	
META-INF	48.2 kB	Folder	
lib	1.5 MB	Folder	
assets	421.1 kB	Folder	
resources.arsc	366.1 kB	unknown	30 November 1979, 00:00
classes.dex	537.3 kB	unknown	30 November 1979, 00:00
AndroidManifest.xml	11.2 kB	XML document	30 November 1979, 00:00

4. This is the point we can view the drawables. The xml and java files are still packaged in the bundle. Viewable Drawables:

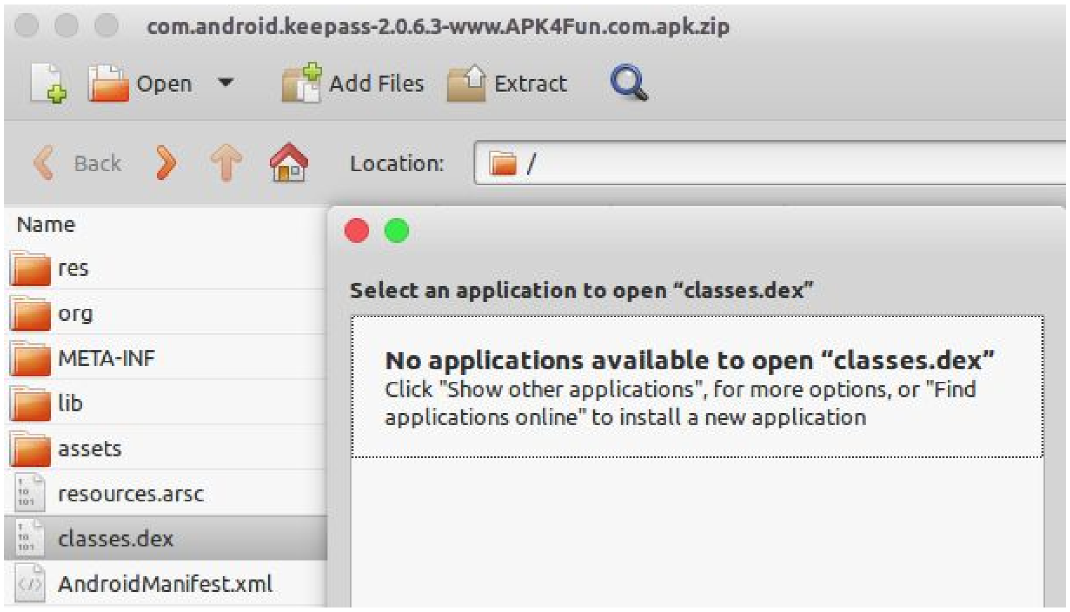


However the .xml file throws a parsing error, i.e not well formatted XML:


com.android.keepass-2.0.6.3-www.APK4Fun.com.apk.zip
Open
Add Files
Extract
Back
Location:
/res/
Name
xml
menu-v11
menu
layout-v11
layout-v4
layout
Mozilla Firefox
file:///home/...earchable.xml
file:///home/pankaj/.cache/.fr-zJtZUI/res/
Search

XML Parsing Error: not well-formed
Location: file:///home/pankaj/.cache/.fr-zJtZUI/res/xml/searchable.xml
Line Number 1, Column 1:

Similarly the Dex file that contains all the java classes show the following status at this point of reverse engineering



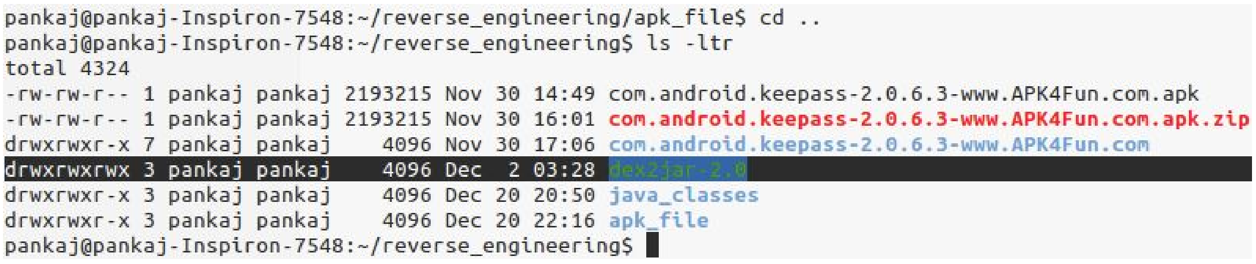
Select an application to open “classes.dex”

No applications available to open “classes.dex”
Click "Show other applications", for more options, or "Find applications online" to install a new application

Step 2:

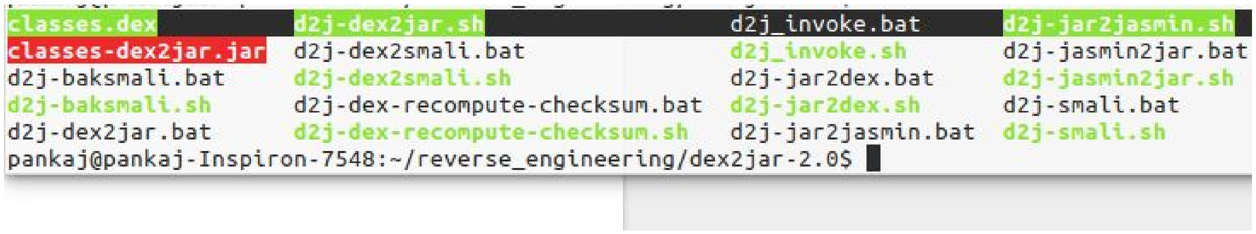
1. Extract the .zip file in the current folder
2. Download dex2jar and extract it to the current folder

3.



```
pankaj@pankaj-Inspiron-7548:~/reverse_engineering/apk_file$ cd ..
pankaj@pankaj-Inspiron-7548:~/reverse_engineering$ ls -ltr
total 4324
-rw-rw-r-- 1 pankaj pankaj 2193215 Nov 30 14:49 com.android.keepass-2.0.6.3-www.APK4Fun.com.apk
-rw-rw-r-- 1 pankaj pankaj 2193215 Nov 30 16:01 com.android.keepass-2.0.6.3-www.APK4Fun.com.apk.zip
drwxrwxr-x 7 pankaj pankaj    4096 Nov 30 17:06 com.android.keepass-2.0.6.3-www.APK4Fun.com
drwxrwxrwx 3 pankaj pankaj    4096 Dec  2 03:28 dex2jar-2.0
drwxrwxr-x 3 pankaj pankaj    4096 Dec 20 20:50 java_classes
drwxrwxr-x 3 pankaj pankaj    4096 Dec 20 22:16 apk_file
pankaj@pankaj-Inspiron-7548:~/reverse_engineering$
```

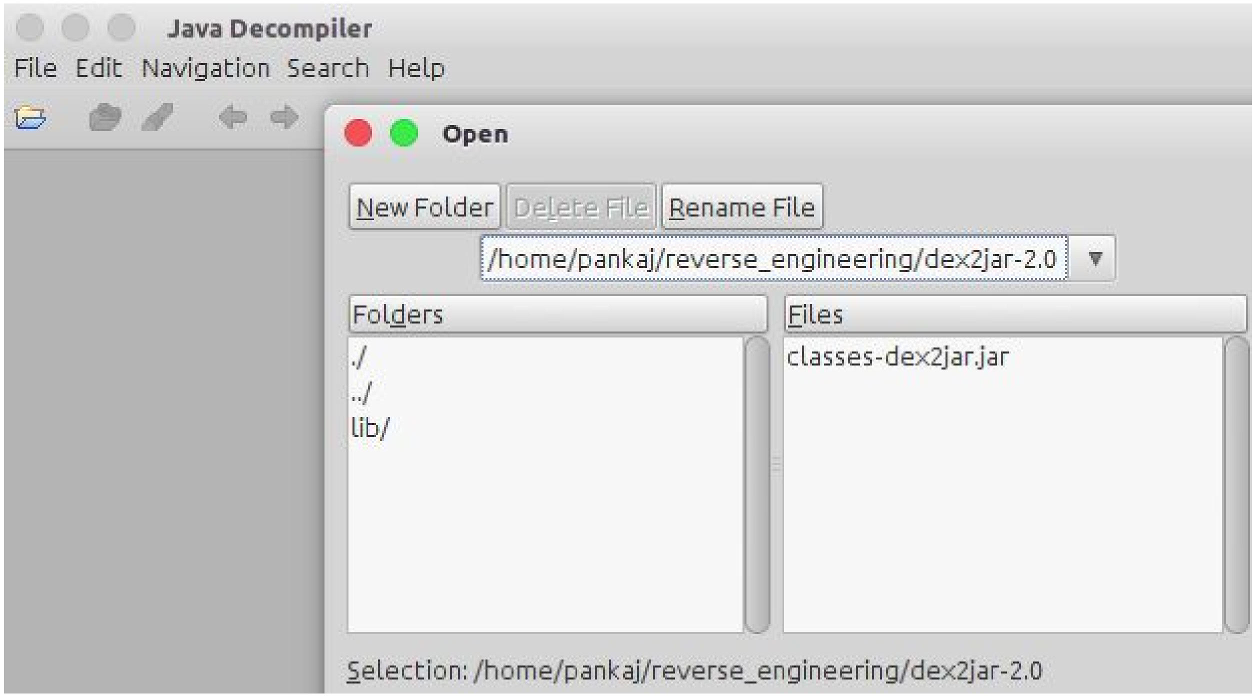
4. Make sure that the classes.dex file and dex2jar lie in the same folder
5. Run this command: `./d2j-dex2jar.sh classes.dex`
6. The above command gives you `classes.dex.dex2jar` file in the same folder.

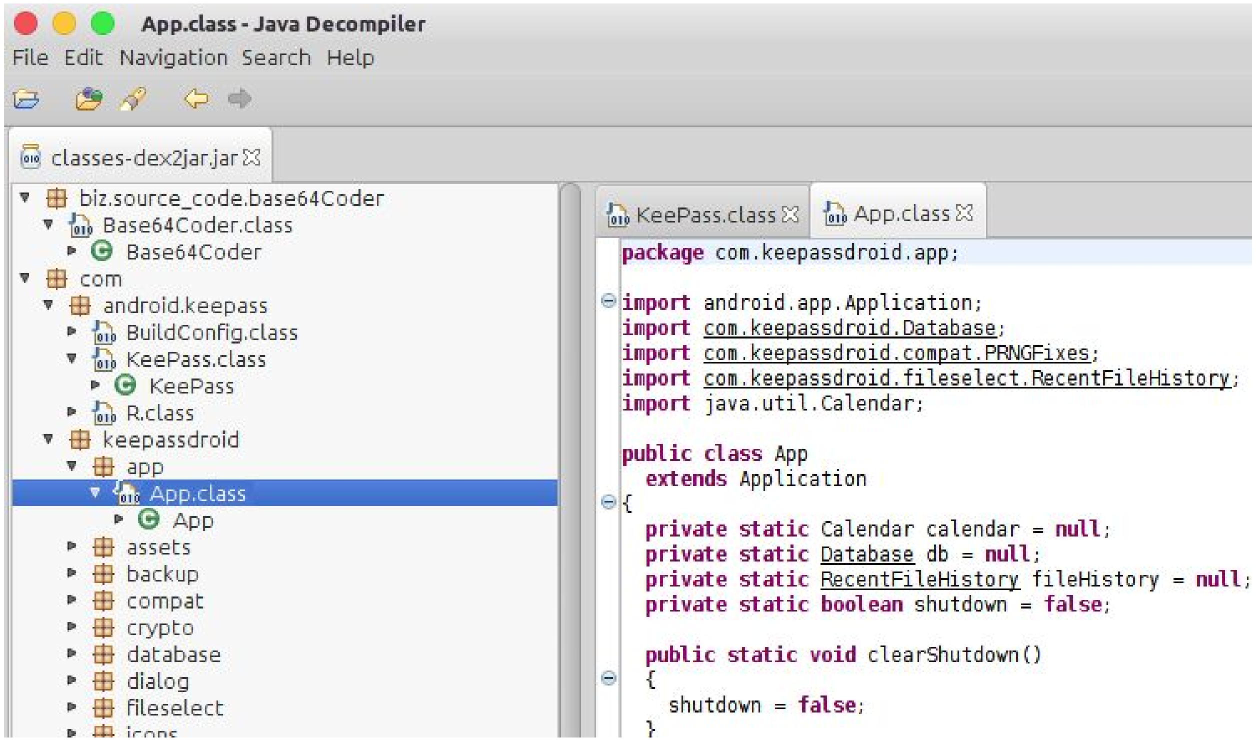


```
classes.dex           d2j-dex2jar.sh                  d2j_invoke.bat       d2j-jar2jasmin.sh
classes-dex2jar.jar   d2j-dex2smali.bat               d2j_invoke.sh        d2j-jasmin2jar.bat
d2j-baksmali.bat      d2j-dex2smali.sh                d2j-jar2dex.bat      d2j-jasmin2jar.sh
d2j-baksmali.sh       d2j-dex-recompute-checksum.bat  d2j-jar2dex.sh       d2j-smali.bat
d2j-dex2jar.bat       d2j-dex-recompute-checksum.sh   d2j-jar2jasmin.bat   d2j-smali.sh
pankaj@pankaj-Inspiron-7548:~/reverse_engineering/dex2jar-2.0$
```

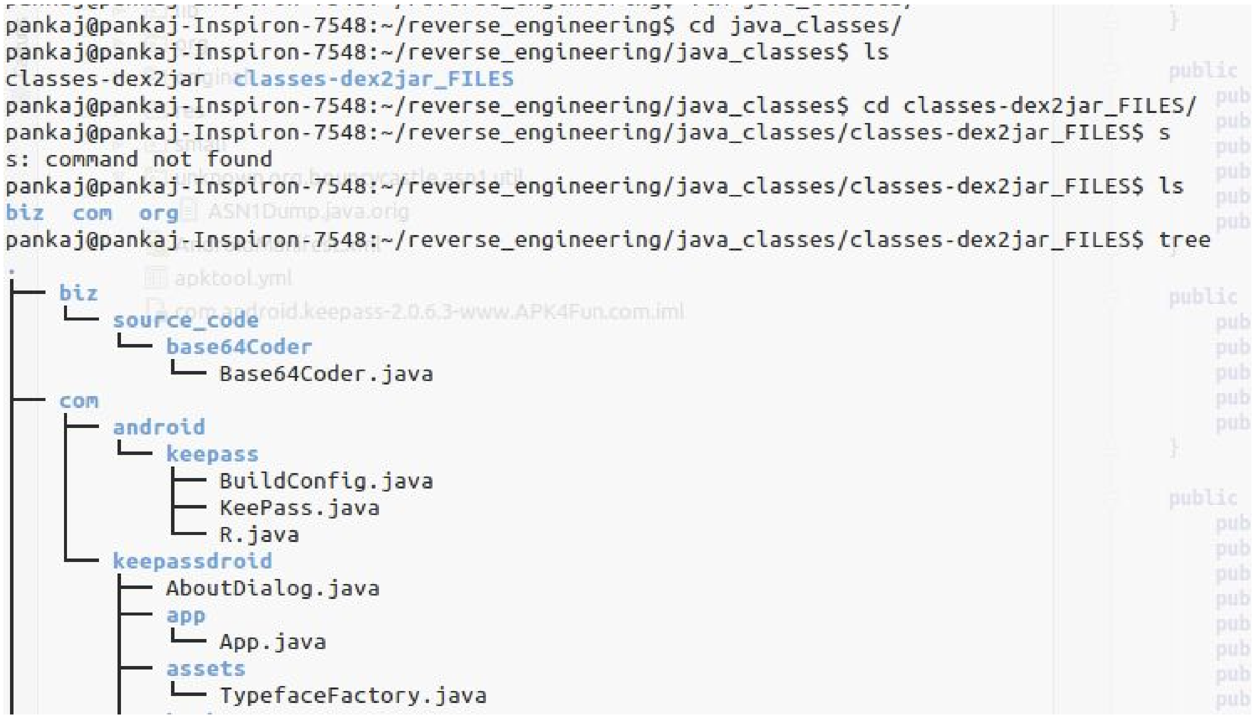
7. Double click on jd-gui after downloading java decompiler. Open classes.dex.dex2jar file from that folder. We have the class files at this point
8. Save all of these class files by source name

   `jd-gui, click File -> Save All Sources`

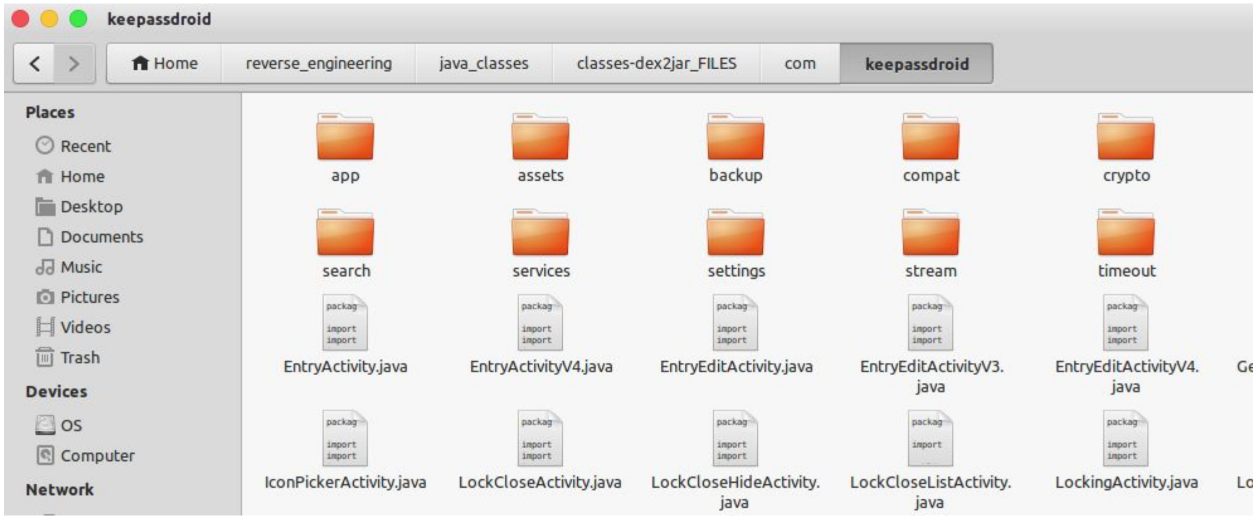

Java Decompiler
File Edit Navigation Search Help
Open
New Folder
Delete File
Rename File
/home/pankaj/reverse_engineering/dex2jar-2.0
Folders
./
../
lib/
Files
classes-dex2jar.jar
Selection: /home/pankaj/reverse_engineering/dex2jar-2.0



```
package com.keepassdroid.app;

import android.app.Application;
import com.keepassdroid.Database;
import com.keepassdroid.compat.PRNGFixes;
import com.keepassdroid.fileselect.RecentFileHistory;
import java.util.Calendar;

public class App
  extends Application
{
  private static Calendar calendar = null;
  private static Database db = null;
  private static RecentFileHistory fileHistory = null;
  private static boolean shutdown = false;

  public static void clearShutdown()
  {
    shutdown = false;
  }
```

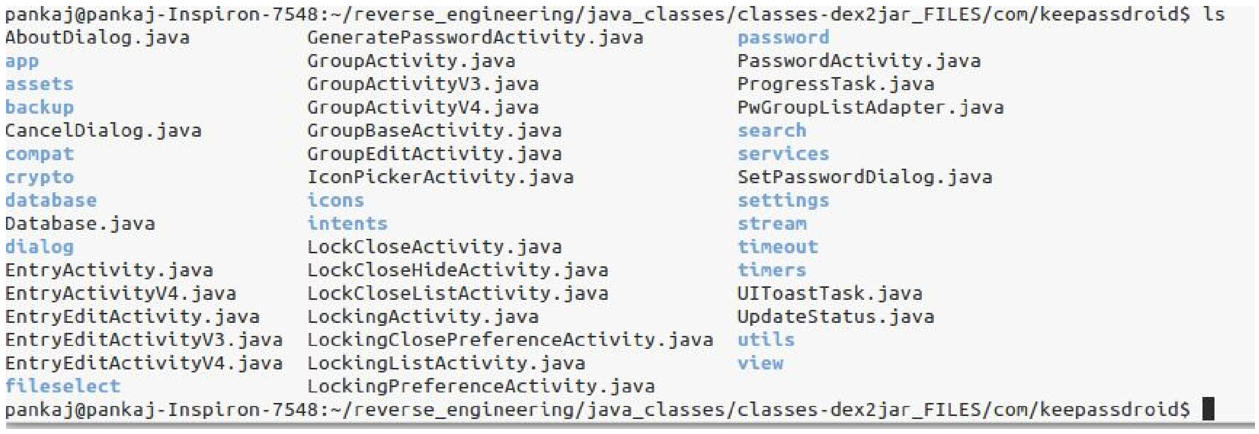


```
pankaj@pankaj-Inspiron-7548:~/reverse_engineering$ cd java_classes/
pankaj@pankaj-Inspiron-7548:~/reverse_engineering/java_classes$ ls
classes-dex2jar  classes-dex2jar_FILES
pankaj@pankaj-Inspiron-7548:~/reverse_engineering/java_classes$ cd classes-dex2jar_FILES/
pankaj@pankaj-Inspiron-7548:~/reverse_engineering/java_classes/classes-dex2jar_FILES$ s
s: command not found
pankaj@pankaj-Inspiron-7548:~/reverse_engineering/java_classes/classes-dex2jar_FILES$ ls
biz  com  org
pankaj@pankaj-Inspiron-7548:~/reverse_engineering/java_classes/classes-dex2jar_FILES$ tree
.
├── biz
│   └── source_code
│       └── base64Coder
│           └── Base64Coder.java
├── com
│   ├── android
│   │   └── keepass
│   │       ├── BuildConfig.java
│   │       ├── KeePass.java
│   │       └── R.java
│   └── keepassdroid
│       ├── AboutDialog.java
│       ├── app
│       │   └── App.java
│       ├── assets
│       │   └── TypefaceFactory.java
```



```
keepassdroid
Home > reverse_engineering > java_classes > classes-dex2jar_FILES > com > keepassdroid

Places: Recent, Home, Desktop, Documents, Music, Pictures, Videos, Trash
Devices: OS, Computer
Network

app                 assets               backup                 compat                   crypto
search              services             settings               stream                   timeout
EntryActivity.java  EntryActivityV4.java EntryEditActivity.java EntryEditActivityV3.java EntryEditActivityV4.java
IconPickerActivity.java LockCloseActivity.java LockCloseHideActivity.java LockCloseListActivity.java LockingActivity.java
```

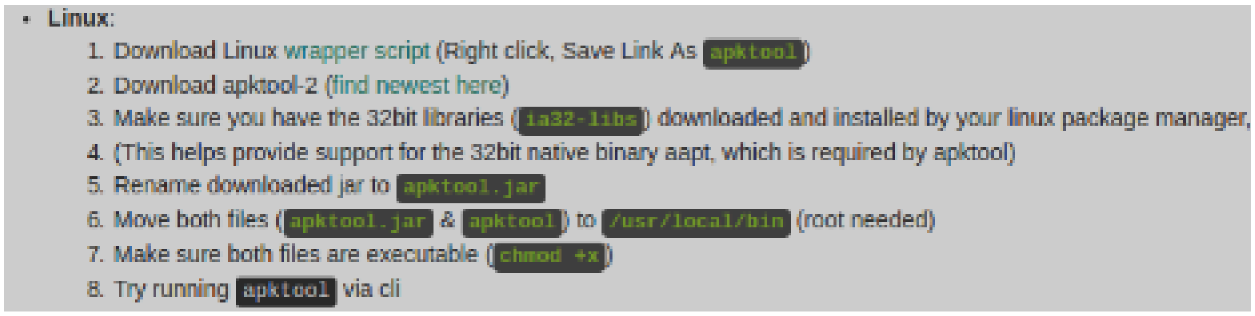
9. At this stage you get the java source but the .xml files are still unreadable, so continue.



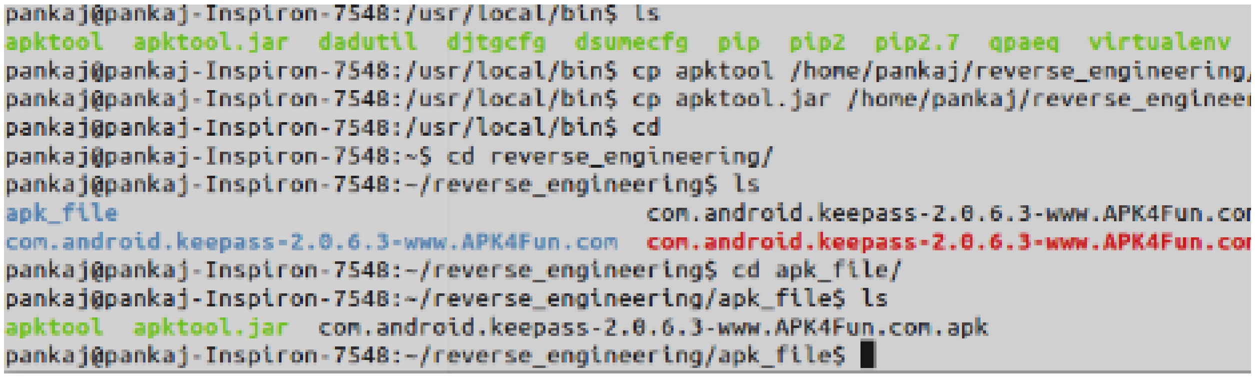
```
pankaj@pankaj-Inspiron-7548:~/reverse_engineering/java_classes/classes-dex2jar_FILES/com/keepassdroid$ ls
AboutDialog.java          GeneratePasswordActivity.java        password
app                       GroupActivity.java                   PasswordActivity.java
assets                    GroupActivityV3.java                 ProgressTask.java
backup                    GroupActivityV4.java                 PwGroupListAdapter.java
CancelDialog.java         GroupBaseActivity.java               search
compat                    GroupEditActivity.java               services
crypto                    IconPickerActivity.java              SetPasswordDialog.java
database                  icons                                settings
Database.java             intents                              stream
dialog                    LockCloseActivity.java               timeout
EntryActivity.java        LockCloseHideActivity.java           timers
EntryActivityV4.java      LockCloseListActivity.java           UIToastTask.java
EntryEditActivity.java    LockingActivity.java                 UpdateStatus.java
EntryEditActivityV3.java  LockingClosePreferenceActivity.java  utils
EntryEditActivityV4.java  LockingListActivity.java             view
fileselect                LockingPreferenceActivity.java
pankaj@pankaj-Inspiron-7548:~/reverse_engineering/java_classes/classes-dex2jar_FILES/com/keepassdroid$
```

Step3:

1. Put in the .apk file which you want to continue with your reverse engineering
2. Download the latest version of apktool and apktool install window (both can be downloaded from the same link) and place them in the same folder



- Linux:
  1. Download Linux wrapper script (Right click, Save Link As `apktool`)
  2. Download apktool-2 (find newest here)
  3. Make sure you have the 32bit libraries (`ia32-libs`) downloaded and installed by your linux package manager,
  4. (This helps provide support for the 32bit native binary aapt, which is required by apktool)
  5. Rename downloaded jar to `apktool.jar`
  6. Move both files (`apktool.jar` & `apktool`) to `/usr/local/bin` (root needed)
  7. Make sure both files are executable (`chmod +x`)
  8. Try running `apktool` via cli

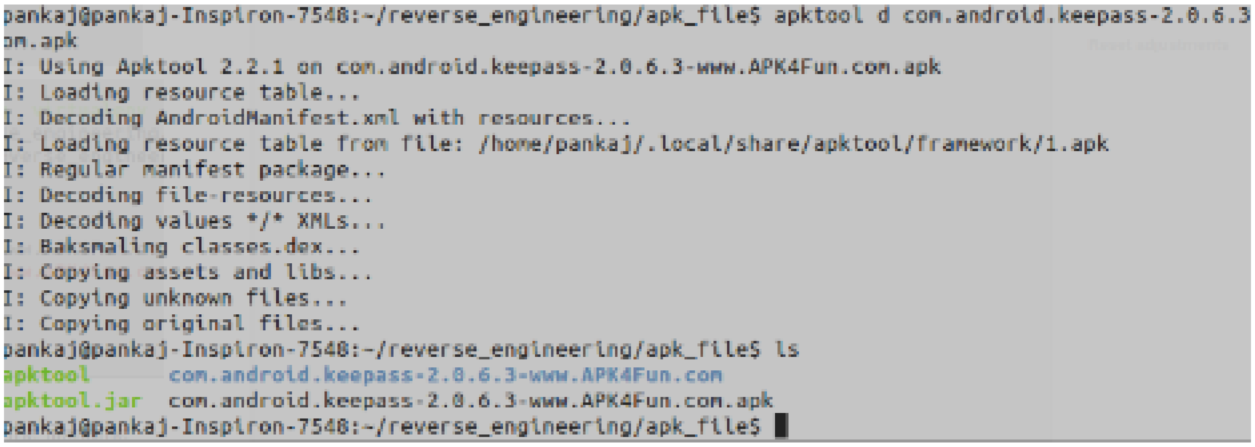


```
pankaj@pankaj-Inspiron-7548:/usr/local/bin$ ls
apktool  apktool.jar  dadutil  djtgcfg  dsumecfg  pip  pip2  pip2.7  qpaeq  virtualenv
pankaj@pankaj-Inspiron-7548:/usr/local/bin$ cp apktool /home/pankaj/reverse_engineering
pankaj@pankaj-Inspiron-7548:/usr/local/bin$ cp apktool.jar /home/pankaj/reverse_enginee
pankaj@pankaj-Inspiron-7548:/usr/local/bin$ cd
pankaj@pankaj-Inspiron-7548:~$ cd reverse_engineering/
pankaj@pankaj-Inspiron-7548:~/reverse_engineering$ ls
apk_file                                    com.android.keepass-2.0.6.3-www.APK4Fun.co
com.android.keepass-2.0.6.3-www.APK4Fun.com  com.android.keepass-2.0.6.3-www.APK4Fun.co
pankaj@pankaj-Inspiron-7548:~/reverse_engineering$ cd apk_file/
pankaj@pankaj-Inspiron-7548:~/reverse_engineering/apk_file$ ls
apktool  apktool.jar  com.android.keepass-2.0.6.3-www.APK4Fun.com.apk
pankaj@pankaj-Inspiron-7548:~/reverse_engineering/apk_file$ 
```

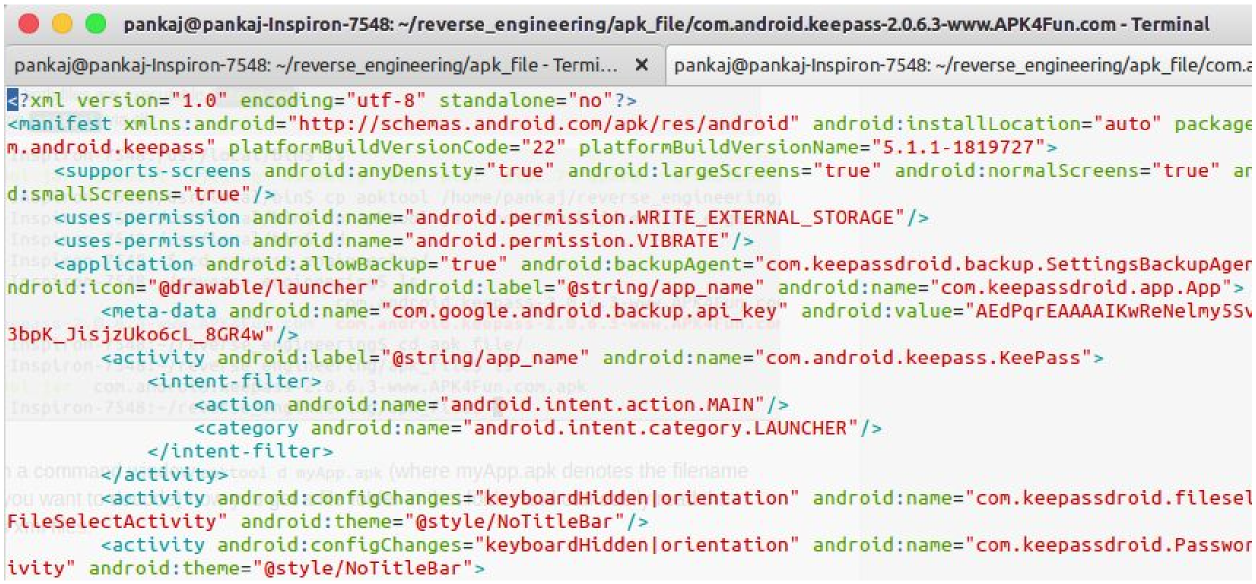
3. Open a command window :

```
apktool d myApp.apk
```

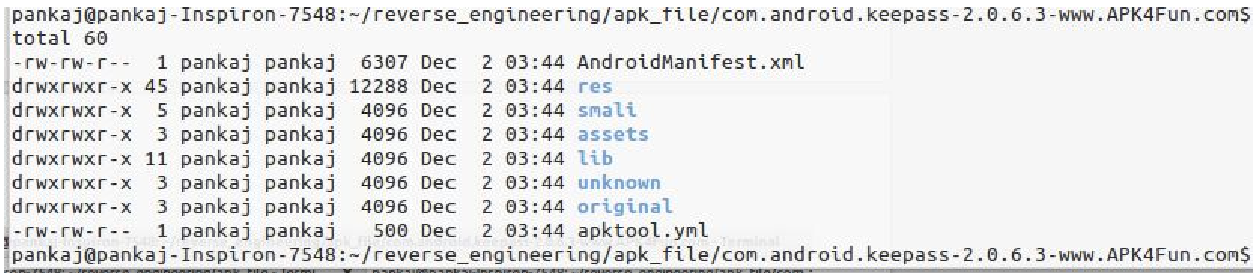
now you can easily read the apk's xml files



```
pankaj@pankaj-Inspiron-7548:~/reverse_engineering/apk_file$ apktool d com.android.keepass-2.0.6.3
om.apk
I: Using Apktool 2.2.1 on com.android.keepass-2.0.6.3-www.APK4Fun.com.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /home/pankaj/.local/share/apktool/framework/1.apk
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...
pankaj@pankaj-Inspiron-7548:~/reverse_engineering/apk_file$ ls
apktool      com.android.keepass-2.0.6.3-www.APK4Fun.com
apktool.jar  com.android.keepass-2.0.6.3-www.APK4Fun.com.apk
pankaj@pankaj-Inspiron-7548:~/reverse_engineering/apk_file$ 
```



```
pankaj@pankaj-Inspiron-7548: ~/reverse_engineering/apk_file/com.android.keepass-2.0.6.3-www.APK4Fun.com - Terminal
pankaj@pankaj-Inspiron-7548: ~/reverse_engineering/apk_file - Termi...  ✕   pankaj@pankaj-Inspiron-7548: ~/reverse_engineering/apk_file/com.a
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:installLocation="auto" package
m.android.keepass" platformBuildVersionCode="22" platformBuildVersionName="5.1.1-1819727">
    <supports-screens android:anyDensity="true" android:largeScreens="true" android:normalScreens="true" an
d:smallScreens="true"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.VIBRATE"/>
    <application android:allowBackup="true" android:backupAgent="com.keepassdroid.backup.SettingsBackupAger
ndroid:icon="@drawable/launcher" android:label="@string/app_name" android:name="com.keepassdroid.app.App">
        <meta-data android:name="com.google.android.backup.api_key" android:value="AEdPqrEAAAAIKwReNelmy5Sv
3bpK_JisjzUko6cL_8GR4w"/>
        <activity android:label="@string/app_name" android:name="com.android.keepass.KeePass">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <activity android:configChanges="keyboardHidden|orientation" android:name="com.keepassdroid.filesel
FileSelectActivity" android:theme="@style/NoTitleBar"/>
        <activity android:configChanges="keyboardHidden|orientation" android:name="com.keepassdroid.Passwor
ivity" android:theme="@style/NoTitleBar">
```

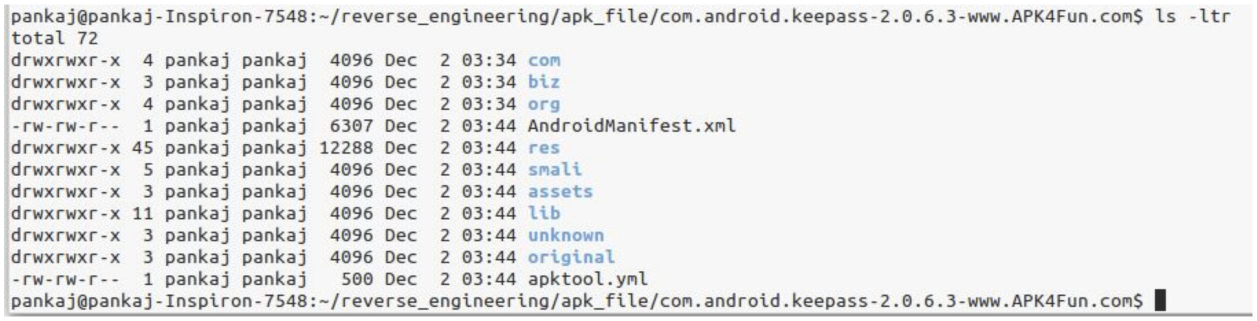


```
pankaj@pankaj-Inspiron-7548:~/reverse_engineering/apk_file/com.android.keepass-2.0.6.3-www.APK4Fun.com$
total 60
-rw-rw-r--  1 pankaj pankaj  6307 Dec  2 03:44 AndroidManifest.xml
drwxrwxr-x 45 pankaj pankaj 12288 Dec  2 03:44 res
drwxrwxr-x  5 pankaj pankaj  4096 Dec  2 03:44 smali
drwxrwxr-x  3 pankaj pankaj  4096 Dec  2 03:44 assets
drwxrwxr-x 11 pankaj pankaj  4096 Dec  2 03:44 lib
drwxrwxr-x  3 pankaj pankaj  4096 Dec  2 03:44 unknown
drwxrwxr-x  3 pankaj pankaj  4096 Dec  2 03:44 original
-rw-rw-r--  1 pankaj pankaj   500 Dec  2 03:44 apktool.yml
pankaj@pankaj-Inspiron-7548:~/reverse_engineering/apk_file/com.android.keepass-2.0.6.3-www.APK4Fun.com$
```

Step 4: Just copy contents of both folder to one folder

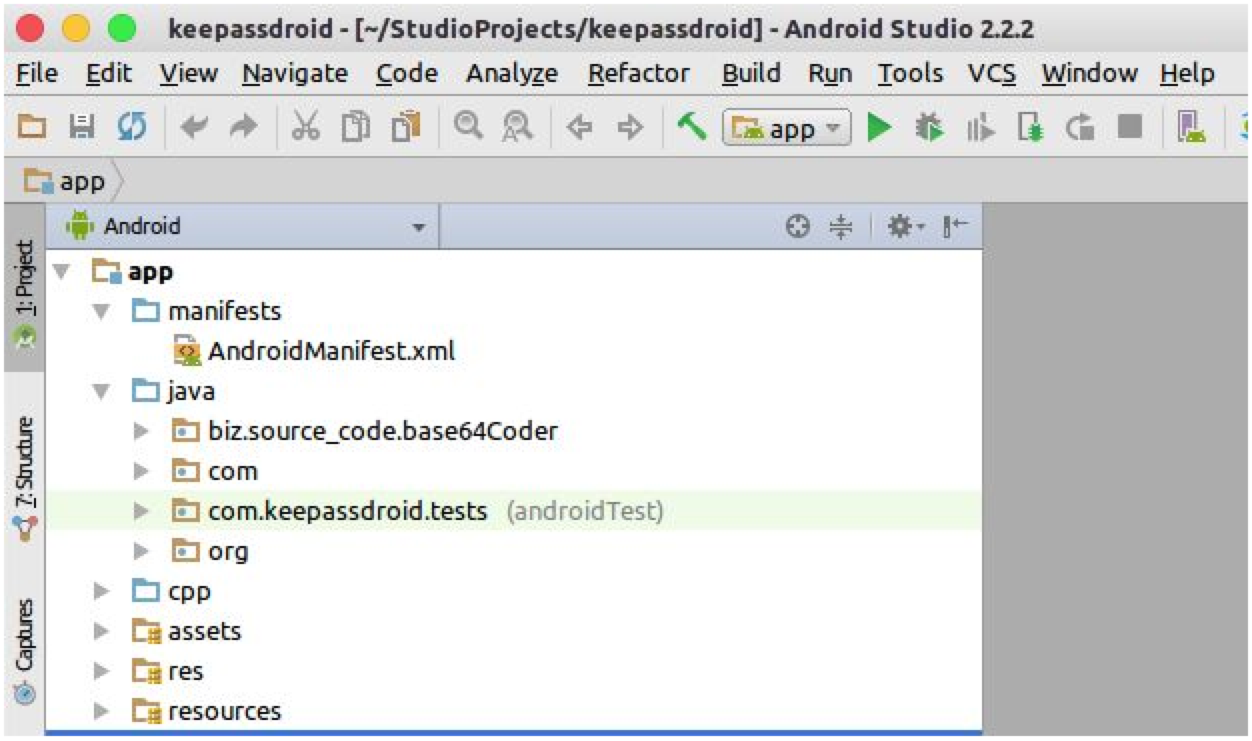
Now you have the source code

Extracted from the reverse engineering process:



```
pankaj@pankaj-Inspiron-7548:~/reverse_engineering/apk_file/com.android.keepass-2.0.6.3-www.APK4Fun.com$ ls -ltr
total 72
drwxrwxr-x  4 pankaj pankaj  4096 Dec  2 03:34 com
drwxrwxr-x  3 pankaj pankaj  4096 Dec  2 03:34 biz
drwxrwxr-x  4 pankaj pankaj  4096 Dec  2 03:34 org
-rw-rw-r--  1 pankaj pankaj  6307 Dec  2 03:44 AndroidManifest.xml
drwxrwxr-x 45 pankaj pankaj 12288 Dec  2 03:44 res
drwxrwxr-x  5 pankaj pankaj  4096 Dec  2 03:44 smali
drwxrwxr-x  3 pankaj pankaj  4096 Dec  2 03:44 assets
drwxrwxr-x 11 pankaj pankaj  4096 Dec  2 03:44 lib
drwxrwxr-x  3 pankaj pankaj  4096 Dec  2 03:44 unknown
drwxrwxr-x  3 pankaj pankaj  4096 Dec  2 03:44 original
-rw-rw-r--  1 pankaj pankaj   500 Dec  2 03:44 apktool.yml
pankaj@pankaj-Inspiron-7548:~/reverse_engineering/apk_file/com.android.keepass-2.0.6.3-www.APK4Fun.com$
```

Actual repository available on Github cloned in android studio:



keepassdroid - [~/StudioProjects/keepassdroid] - Android Studio 2.2.2

```
app
  manifests
    AndroidManifest.xml
  java
    biz.source_code.base64Coder
    com
    com.keepassdroid.tests (androidTest)
    org
  cpp
  assets
  res
  resources
```

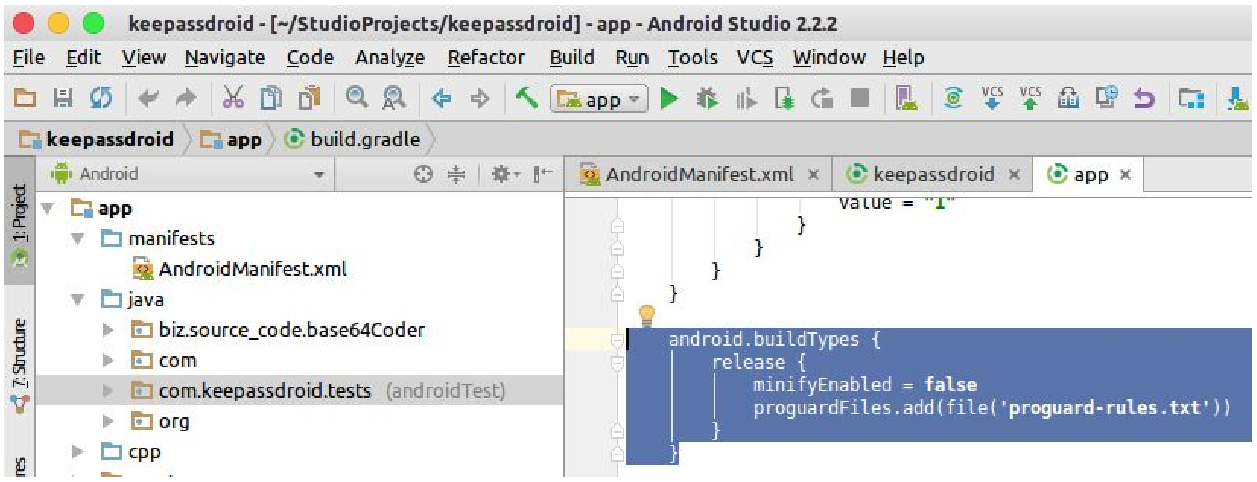
From the above two screenshots you can see all the relevant files are cracked from the APK

Key Findings / Recommendations

Proguard

Android developer's guide enables developers to remove unused code and resources during deployment of the build. This helps to shrink the APK file and reduces its size which matters a lot. This is possible with android development using Proguard in gradle settings as shown in the Figure below. Proguard shrinks the code by detecting and omitting the unused classes, methods, attributes, fields, from the packaged app. It also shrinks those classes and fields included from libraries.

Proguard actually optimizes the bytecode. With proguard guarding your project, the code gets obfuscated which even makes android APK difficult to reverse engineer. This proves to be an excellent option when your app uses encryption algorithms to store confidential data or security-sensitive features, such as licensing verification. Following is a screenshot from the repository code of keepassdroid simulated in my local machine on Android studio.


keepassdroid - [~/StudioProjects/keepassdroid] - app - Android Studio 2.2.2
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
keepassdroid app build.gradle
Android
AndroidManifest.xml keepassdroid app
1: Project
7: Structure
app
manifests
AndroidManifest.xml
java
biz.source_code.base64Coder
com
com.keepassdroid.tests (androidTest)
org
cpp

```
                value = "1"
            }
        }
    }
}
android.buildTypes {
    release {
        minifyEnabled = false
        proguardFiles.add(file('proguard-rules.txt'))
    }
}
```

Database encryption for keepass:

Following block ciphers are used to encrypt the databases. Everything that goes in the DB gets encrypted along with the password:

Cipher	Block Size	Key size
AES	128bits	256bits

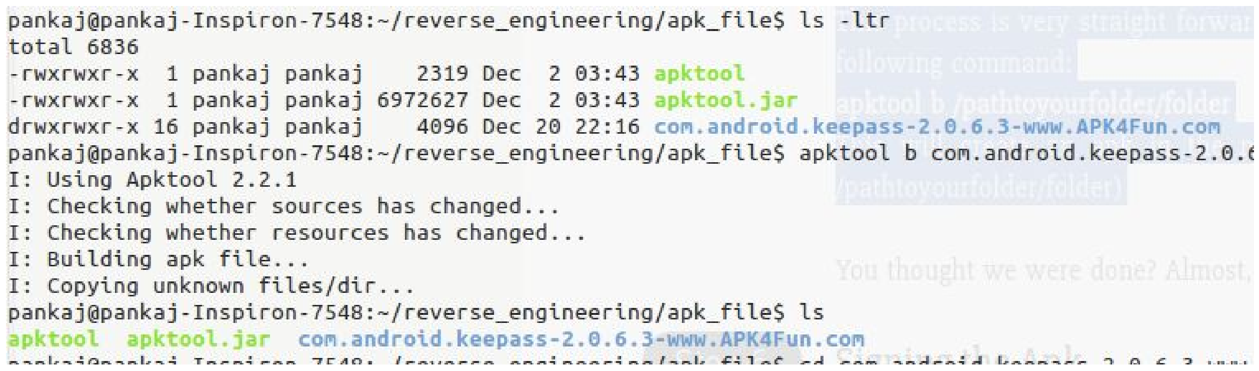
Security issues for keepass [10]

- Header Authentication: Keepass uses KDB and KDBX file formats. Header Authentication for KDB, indulges silent data removal attacks whereas for KDBX silent data corruption attacks is possible. However, both are minor security issues
- The method, “MemUtil.ArraysEqual” is susceptible to Timing Attack

Further Points to android Reverse Engineering

Compiling Your App from the java class you extracted.

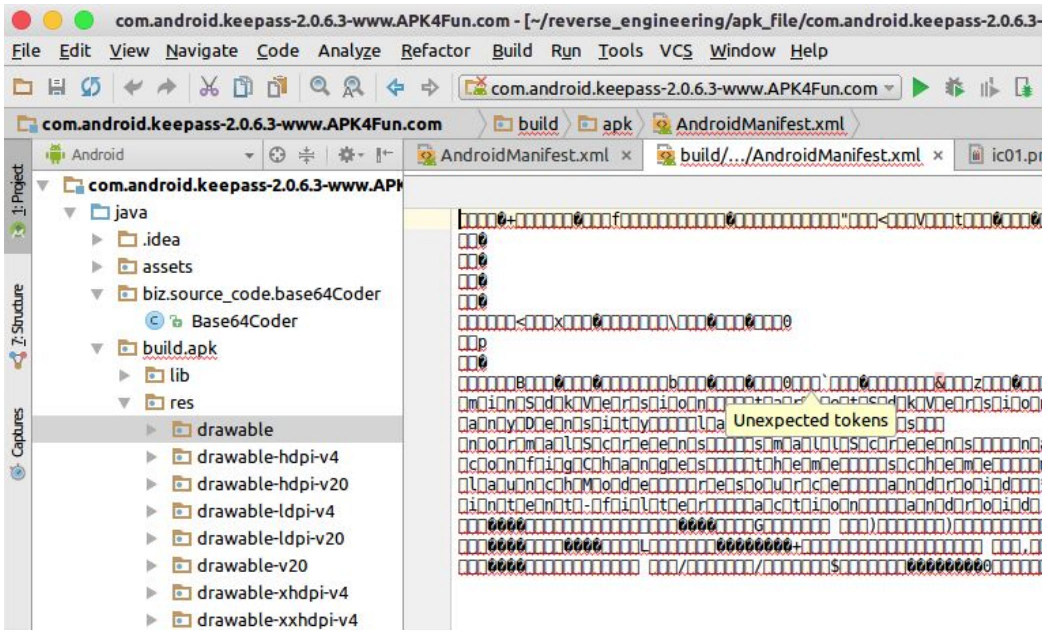
```
apktool b /pathtoyourfolder/folder
```



```
pankaj@pankaj-Inspiron-7548:~/reverse_engineering/apk_file$ ls -ltr
total 6836
-rwxrwxr-x  1 pankaj pankaj    2319 Dec  2 03:43 apktool
-rwxrwxr-x  1 pankaj pankaj 6972627 Dec  2 03:43 apktool.jar
drwxrwxr-x 16 pankaj pankaj    4096 Dec 20 22:16 com.android.keepass-2.0.6.3-www.APK4Fun.com
pankaj@pankaj-Inspiron-7548:~/reverse_engineering/apk_file$ apktool b com.android.keepass-2.0.6
I: Using Apktool 2.2.1
I: Checking whether sources has changed...
I: Checking whether resources has changed...
I: Building apk file...
I: Copying unknown files/dir...
pankaj@pankaj-Inspiron-7548:~/reverse_engineering/apk_file$ ls
apktool  apktool.jar  com.android.keepass-2.0.6.3-www.APK4Fun.com
```

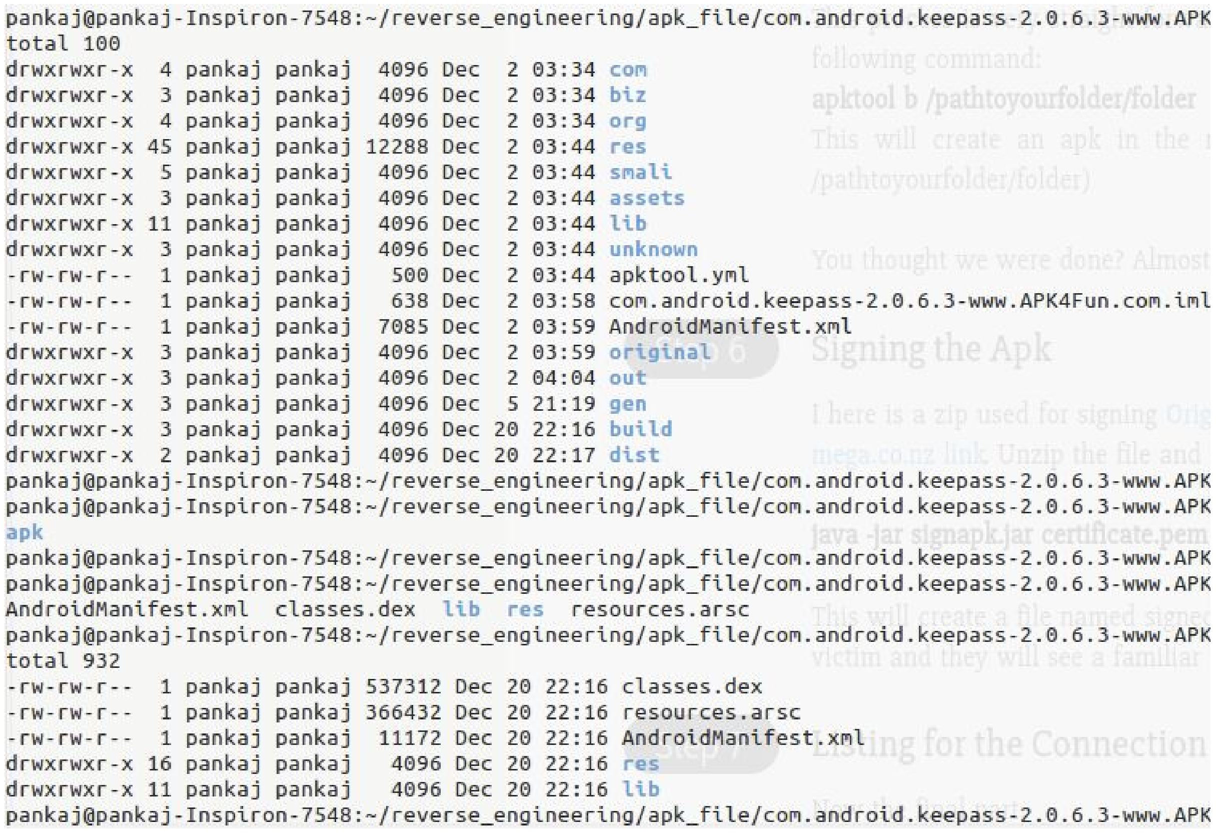
```
pankaj@pankaj-Inspiron-7548:~/reverse_engineering/apk_file$
apktool b com.android.keepass-2.0.6.3-www.APK4Fun.com
I: Using Apktool 2.2.1
I: Checking whether sources has changed…
I: Checking whether resources has changed...
I: Building apk file...
I: Copying unknown files/dir…
```

AndroidManifest.xml file of the generated apk, by above process


com.android.keepass-2.0.6.3-www.APK4Fun.com - [~/reverse_engineering/apk_file/com.android.keepass-2.0.6.3-
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
com.android.keepass-2.0.6.3-www.APK4Fun.com
com.android.keepass-2.0.6.3-www.APK4Fun.com
build
apk
AndroidManifest.xml
Android
AndroidManifest.xml
build/.../AndroidManifest.xml
ic01.p
1: Project
7: Structure
Captures
com.android.keepass-2.0.6.3-www.APK
java
.idea
assets
biz.source_code.base64Coder
Base64Coder
build.apk
lib
res
drawable
drawable-hdpi-v4
drawable-hdpi-v20
drawable-ldpi-v4
drawable-ldpi-v20
drawable-v20
drawable-xhdpi-v4
drawable-xxhdpi-v4
Unexpected tokens

You can see the /build/apk folder being created in the screenshot:

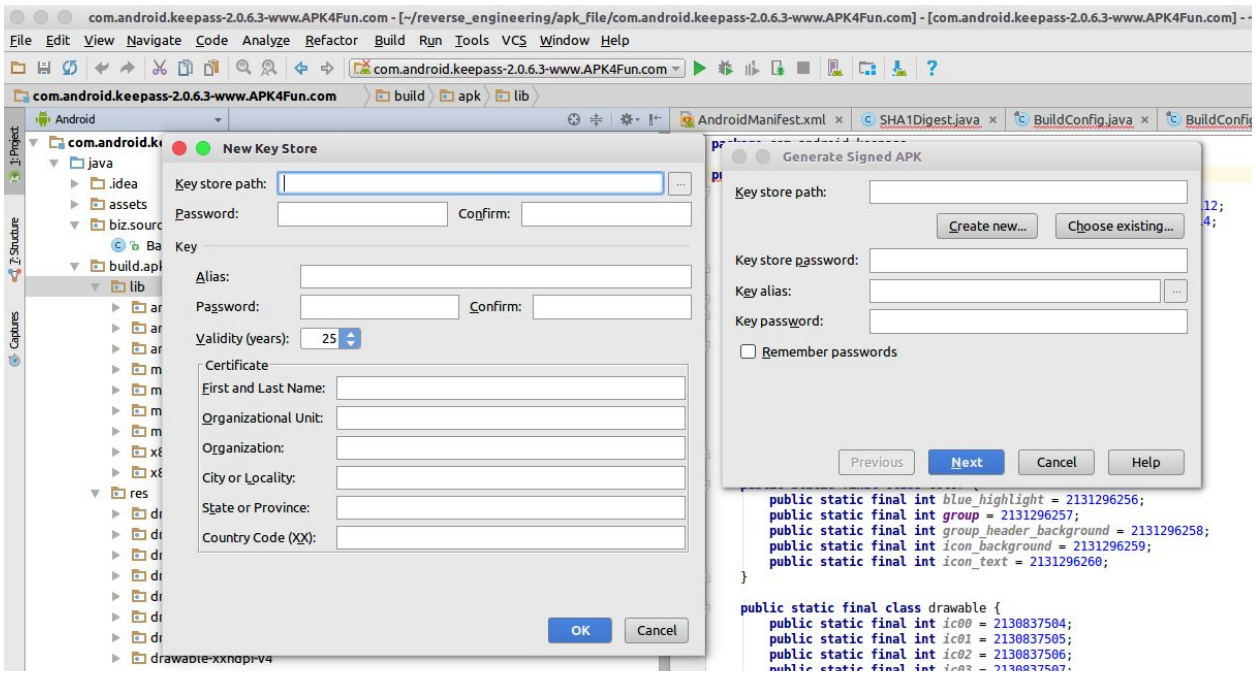
```
pankaj@pankaj-Inspiron-7548:~/reverse_engineering/apk_file/com.android.keepass
-2.0.6.3-www.APK4Fun.com/build/apk$ ls -ltr
total 932
-rw-rw-r--  1 pankaj pankaj 537312 Dec 20 22:16 classes.dex
-rw-rw-r--  1 pankaj pankaj 366432 Dec 20 22:16 resources.arsc
-rw-rw-r--  1 pankaj pankaj  11172 Dec 20 22:16 AndroidManifest.xml
drwxrwxr-x 16 pankaj pankaj   4096 Dec 20 22:16 res
drwxrwxr-x 11 pankaj pankaj   4096 Dec 20 22:16 lib
```



```
pankaj@pankaj-Inspiron-7548:~/reverse_engineering/apk_file/com.android.keepass-2.0.6.3-www.APK
total 100
drwxrwxr-x  4 pankaj pankaj  4096 Dec  2 03:34 com
drwxrwxr-x  3 pankaj pankaj  4096 Dec  2 03:34 biz
drwxrwxr-x  4 pankaj pankaj  4096 Dec  2 03:34 org
drwxrwxr-x 45 pankaj pankaj 12288 Dec  2 03:44 res
drwxrwxr-x  5 pankaj pankaj  4096 Dec  2 03:44 smali
drwxrwxr-x  3 pankaj pankaj  4096 Dec  2 03:44 assets
drwxrwxr-x 11 pankaj pankaj  4096 Dec  2 03:44 lib
drwxrwxr-x  3 pankaj pankaj  4096 Dec  2 03:44 unknown
-rw-rw-r--  1 pankaj pankaj   500 Dec  2 03:44 apktool.yml
-rw-rw-r--  1 pankaj pankaj   638 Dec  2 03:58 com.android.keepass-2.0.6.3-www.APK4Fun.com.iml
-rw-rw-r--  1 pankaj pankaj  7085 Dec  2 03:59 AndroidManifest.xml
drwxrwxr-x  3 pankaj pankaj  4096 Dec  2 03:59 original
drwxrwxr-x  3 pankaj pankaj  4096 Dec  2 04:04 out
drwxrwxr-x  3 pankaj pankaj  4096 Dec  5 21:19 gen
drwxrwxr-x  3 pankaj pankaj  4096 Dec 20 22:16 build
drwxrwxr-x  2 pankaj pankaj  4096 Dec 20 22:17 dist
pankaj@pankaj-Inspiron-7548:~/reverse_engineering/apk_file/com.android.keepass-2.0.6.3-www.APK
pankaj@pankaj-Inspiron-7548:~/reverse_engineering/apk_file/com.android.keepass-2.0.6.3-www.APK
apk
pankaj@pankaj-Inspiron-7548:~/reverse_engineering/apk_file/com.android.keepass-2.0.6.3-www.APK
pankaj@pankaj-Inspiron-7548:~/reverse_engineering/apk_file/com.android.keepass-2.0.6.3-www.APK
AndroidManifest.xml  classes.dex  lib  res  resources.arsc
pankaj@pankaj-Inspiron-7548:~/reverse_engineering/apk_file/com.android.keepass-2.0.6.3-www.APK
total 932
-rw-rw-r--  1 pankaj pankaj 537312 Dec 20 22:16 classes.dex
-rw-rw-r--  1 pankaj pankaj 366432 Dec 20 22:16 resources.arsc
-rw-rw-r--  1 pankaj pankaj  11172 Dec 20 22:16 AndroidManifest.xml
drwxrwxr-x 16 pankaj pankaj   4096 Dec 20 22:16 res
drwxrwxr-x 11 pankaj pankaj   4096 Dec 20 22:16 lib
pankaj@pankaj-Inspiron-7548:~/reverse_engineering/apk_file/com.android.keepass-2.0.6.3-www.APK
```

Signing the Apk

Given the reverse engineered project is compiled, following Figure shows a way to generate a signed apk, which marks the last step of reverse engineering of an android app.


com.android.keepass-2.0.6.3-www.APK4Fun.com - [~/reverse_engineering/apk_file/com.android.keepass-2.0.6.3-www.APK4Fun.com] - [com.android.keepass-2.0.6.3-www.APK4Fun.com]
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
com.android.keepass-2.0.6.3-www.APK4Fun.com
build
apk
lib
Android
AndroidManifest.xml
SHA1Digest.java
BuildConfig.java
java
.idea
assets
res
drawable-xxhdpi-v4
OK
Cancel
Create new...
Choose existing...
Previous
Next
Cancel
Help

New Key Store

Key store path:
Password: Confirm:
Key
Alias:
Password: Confirm:
Validity (years): 25
Certificate
First and Last Name:
Organizational Unit:
Organization:
City or Locality:
State or Province:
Country Code (XX):

Generate Signed APK

Key store path:
Key store password:
Key alias:
Key password:
Remember passwords

```
    public static final int blue_highlight = 2131296256;
    public static final int group = 2131296257;
    public static final int group_header_background = 2131296258;
    public static final int icon_background = 2131296259;
    public static final int icon_text = 2131296260;
}

public static final class drawable {
    public static final int ic00 = 2130837504;
    public static final int ic01 = 2130837505;
    public static final int ic02 = 2130837506;
```

Note: (Because the project is proguard obfuscated, generating an apk is not feasible)

However, here's a command to generate the signed apk, given you have the *.jar and .*pem[6]

```
java -jar *.jar *.pem key.pk8 /pathtoyourapp/app.apk *.apk
```
[6]

This creates an apk file.

Summary

As the application developer has worked very hard for obfuscating code, proguard is likely to be enabled, like in the case of keepassdroid as mentioned above.

So it won't be easy to crack the files required to build an apk from the reverse engineered code due to missing chunks, specially gradle files.

Files like java, res and xml were reverse engineered as seen from the screenshots.

Further work, to complete reverse engineering process includes creating an APK from the cracked code. Then creating a signed build using the *.jar and self-signed keys. However it won't be possible if the code is obfuscated.

Currently android leads the JAVA and Linux platform. Reverse engineering the app comes easy when the resources are available for free like android and LInux as opposed to other platforms dependent application. Reverse engineering can be made complex for hackers/malicious app developers by applying security approaches towards development. Such security practices change both the outlooks, dynamic as well as static of the application. Worst case, if the application/software is dealing with intense confident data like something relevant to banking, government, defence, education or some payment gateway integration then an ideal scenario shall be hosting all of the information on server and pulling the data from the servers through secure APIs.

References:

[1] Android, Google play Licensing (n.d.)

Retrieved from Android:

https://developer.android.com/google/play/licensing/overview.html

[2] APK, Wikipedia (n.d.)

Retrieved from Wikipedia:

https://en.wikipedia.org/wiki/Android_application_package

[3] Sudipta Ghosh, S. R. Tandan, Kamlesh Lahre (June - 2013)

Retrieved from ijert.org:

https://www.ijert.org/view-pdf/4095/shielding-android-application-against-reverse-engin
eering

[4] Google play store, Keepassdroid (n.d.)

Retrieved from Play Store:

https://play.google.com/store/apps/details?id=com.android.keepass&hl=en

[5] metasploit-framework, Github (n.d.)

Retrieved from Github:

https://github.com/rapid7/metasploit-framework/wiki/How-to-use-msfvenom

[6] Forum XDA developer(30th March 2013)

Retrieved from Forum XDA:

http://forum.xda-developers.com/showthread.php?t=2213985

[7] Apktool, ibotpeaches(n.d.)

Retrieved from Apktool: https://ibotpeaches.github.io/Apktool/install/

[8] Dex2jar, github(n.d.)

Retrieved from Github: https://github.com/pxb1988/dex2jar

[9] JD-GUI, benow(n.d.)

Retrieved from benow: http://jd.benow.ca/

[10] Keepass(n.d.)

Retrieved from Keepass: http://keepass.info/help/kb/sec_issues.html