Apkpatcher

Workshop on application patching.

Benoît FORGETTE (MadSquirrels)

05/06/2025

Quarkslab

Who am I ?

Benoît FORGETTE

Software and hardware Security Researcher

topic (Hardware/Android)

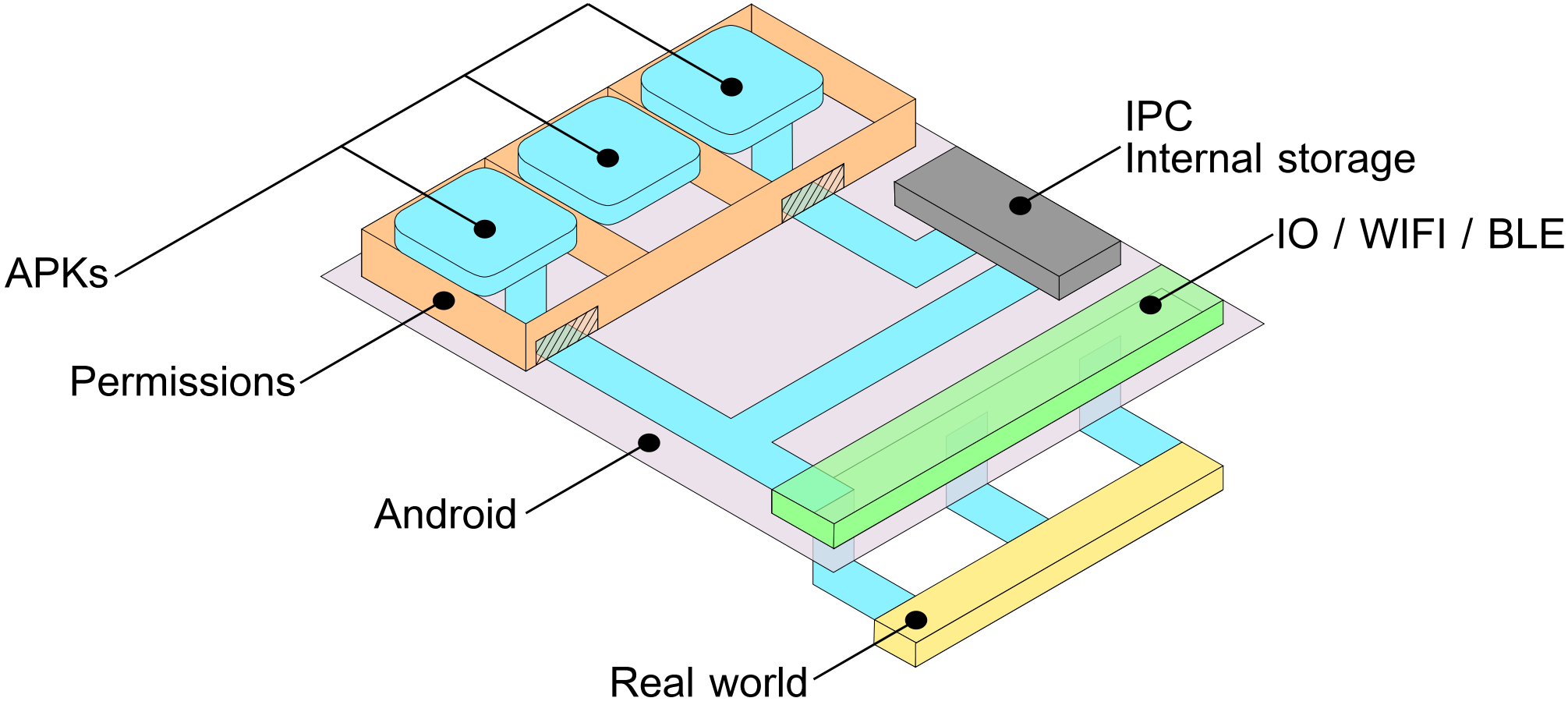
Link for challenge

https://transfer.ci-yow.com/d?id=zPsd93uWjSmjALn

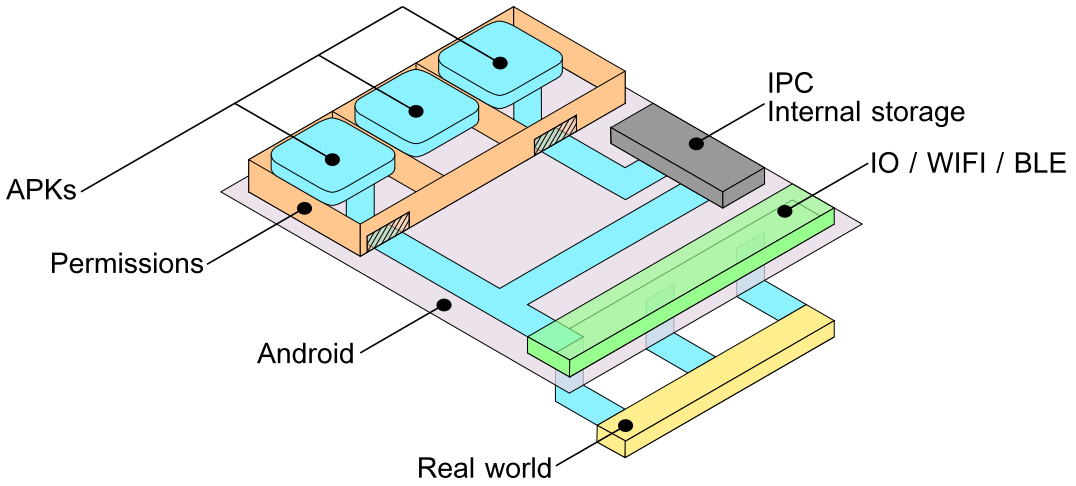
Let's go back to basics.

Quarkslab

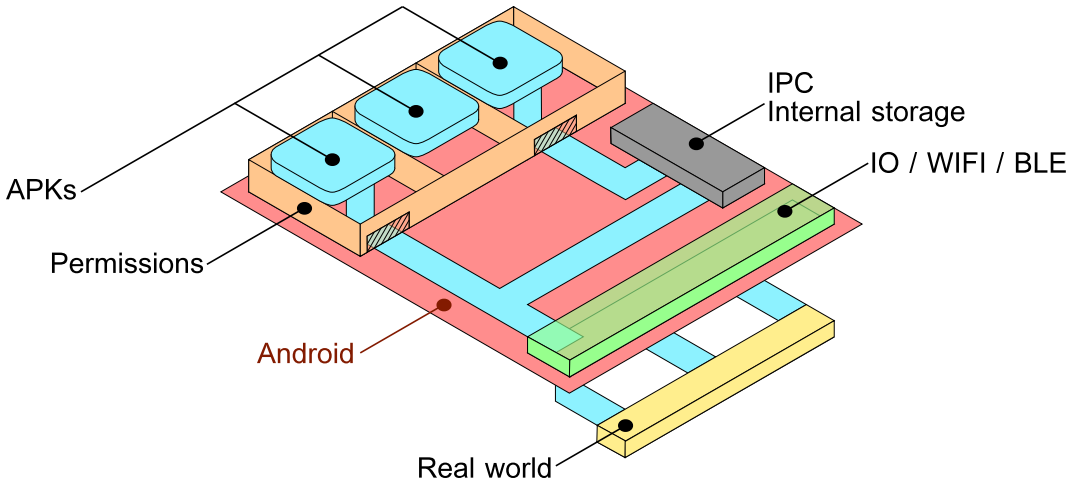
Why we would like to patch



Root an Android

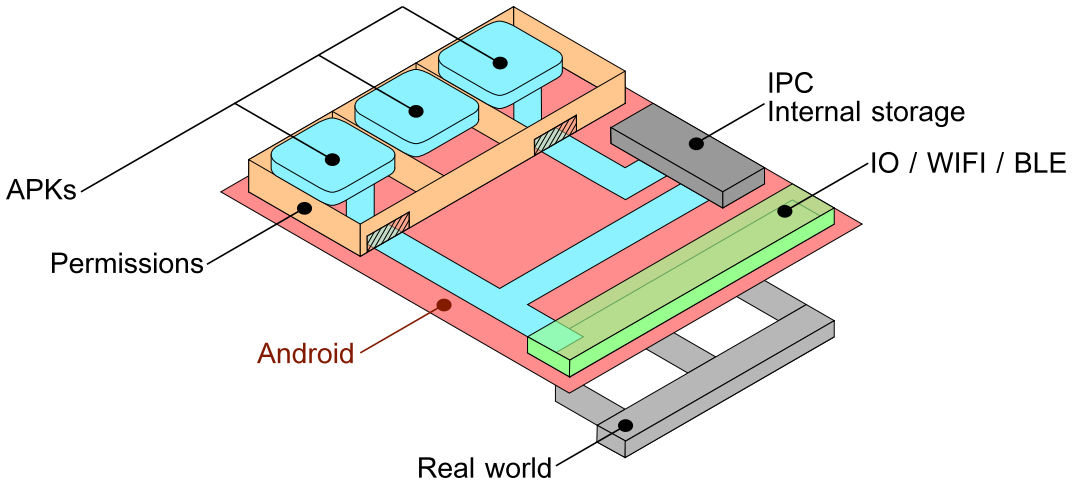


Root an Android



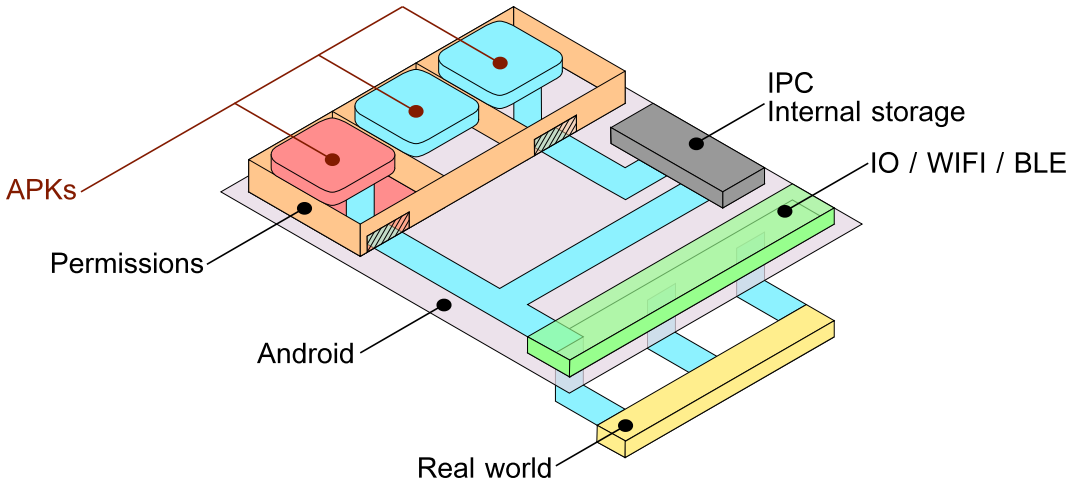
- ❌ A full chain exploitation is needed;
- ❌ It is more commonly know and integrated on detection mechanism;
- ❌ Could not be able to deliver a new APK with modified behavior;
- ✅ Have a full access on system.

Emulate Android



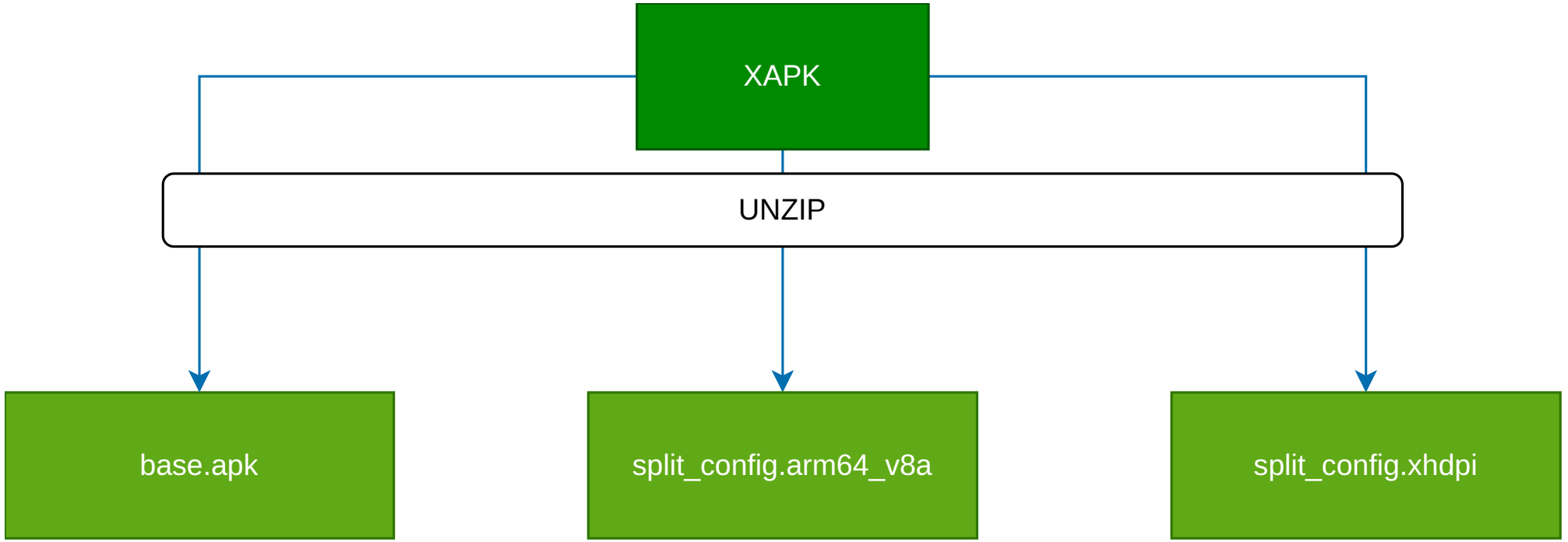
- ❌ Is more commonly know and integrated on detection mechanism;
- ❌ Could not be able to deliver a new APK with modified behavior;
- ❌ Do not have not a full access to external feature;
- ❌ Limited to OS architecture (X86);
- ✅ Have a full access on system.

Patched an APK

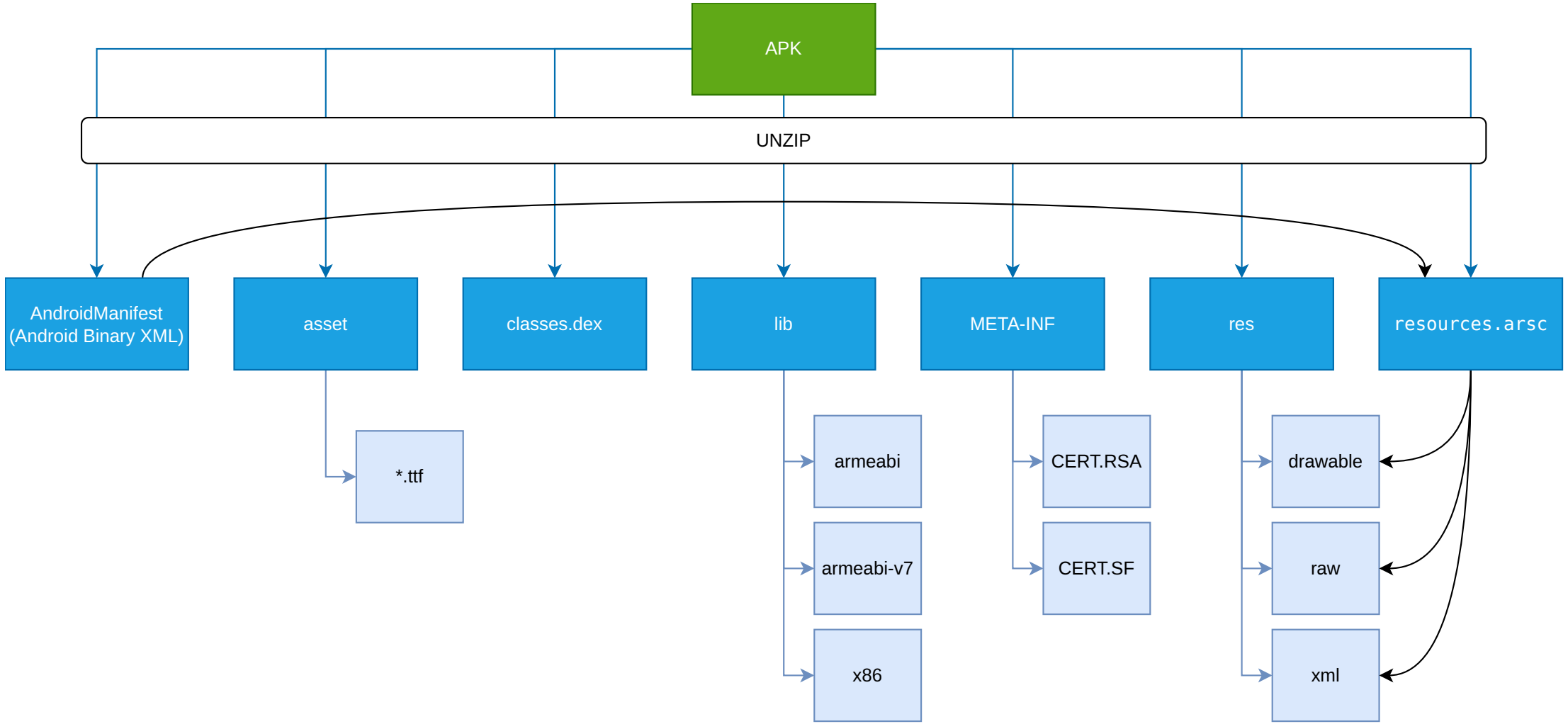


- ❌ Could not by-pass Package manager if a vendor certificate is setup;
- ❌ Need to have an ADB access on Android target;
- ✅ Enable to modify the apk;
- ✅ Can works on any Android system.

Different format



Application structure



Demo unpack

Only unpack or repack

```
apkpatcher -a <apk> --only-unpack dir
apkpatcher -a <new_apk> --only-repack dir
```

Application structure

AndroidManifest

```
<manifest package="com.example.pro" platformBuildVersionCode="34" platformBuildVersionName="14">
  <uses-permission android:name="android.permission.ACTIVITY_RECOGNITION"/>
  <application android:theme="@7F16059E" android:label="@7F150398" android:icon="@7F110000" android:debuggable="true"
    android:name="com.fitnesskeeper.runkeeper.RunKeeperApplication"
    android:networkSecurityConfig="@7F18000C" >
    <meta-data android:name="com.google.firebase.messaging.default_notification_icon" android:resource="@7F080413"/>
    <meta-data android:name="com.google.firebase.messaging.default_notification_color" android:resource="@7F06035C"/>
    <activity android:theme="@7F16052C" android:name="com.fitnesskeeper.runkeeper.SplashActivity" android:exported="true">
      <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.LAUNCHER"/>
      </intent-filter>
    </activity>
  </application>
</manifest>
```

resources.arsc

```
<public type="mipmap" name="ic_launcher" id="0x7f110000" data="res/9w.png" data_size=8/>
<public type="mipmap" name="ic_launcher" id="0x7f110000" data="res/yn.png" data_size=8/>
<public type="mipmap" name="ic_launcher" id="0x7f110000" data="res/FS.png" data_size=8/>
<public type="mipmap" name="ic_launcher" id="0x7f110000" data="res/RJ.png" data_size=8/>
<public type="mipmap" name="ic_launcher" id="0x7f110000" data="res/o-.png" data_size=8/>
<public type="mipmap" name="ic_launcher" id="0x7f110000" data="res/BW.xml" data_size=8/>
<public type="string" name="global_celsius" id="0x7f150398" data="C" data_size=8/>
<public type="style" name="TripSummary.Card.NoMapNoPhoto.Label" id="0x7f16059e" data="0x0" data_size=0/>
<public type="xml" name="network_security_config" id="0x7f18000c" data="res/8G.xml" data_size=8/>
```

External resources is identified by an id reused in resources.arsc

Application structure

AndroidManifest

```
<manifest package="com.example.pro" platformBuildVersionCode="34" platformBuildVersionName="14">
  <uses-permission android:name="android.permission.ACTIVITY_RECOGNITION"/>
  <application android:theme="@7F16059E" android:label="@7F150398" android:icon="@7F110000" android:debuggable="true"
    android:name="com.fitnesskeeper.runkeeper.RunKeeperApplication"
    android:networkSecurityConfig="@7F18000C" >
    <meta-data android:name="com.google.firebase.messaging.default_notification_icon" android:resource="@7F080413"/>
    <meta-data android:name="com.google.firebase.messaging.default_notification_color" android:resource="@7F06035C"/>
    <activity android:theme="@7F16052C" android:name="com.fitnesskeeper.runkeeper.SplashActivity" android:exported="true">
      <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.LAUNCHER"/>
      </intent-filter>
    </activity>
  </application>
</manifest>
```

resources.arsc

```
<public type="mipmap" name="ic_launcher" id="0x7f110000" data="res/9w.png" data_size=8/>
<public type="mipmap" name="ic_launcher" id="0x7f110000" data="res/yn.png" data_size=8/>
<public type="mipmap" name="ic_launcher" id="0x7f110000" data="res/FS.png" data_size=8/>
<public type="mipmap" name="ic_launcher" id="0x7f110000" data="res/RJ.png" data_size=8/>
<public type="mipmap" name="ic_launcher" id="0x7f110000" data="res/o-.png" data_size=8/>
<public type="mipmap" name="ic_launcher" id="0x7f110000" data="res/BW.xml" data_size=8/>
<public type="string" name="global_celsius" id="0x7f150398" data="C" data_size=8/>
<public type="style" name="TripSummary.Card.NoMapNoPhoto.Label" id="0x7f16059e" data="0x0" data_size=0/>
<public type="xml" name="network_security_config" id="0x7f18000c" data="res/8G.xml" data_size=8/>
```

External resources is identified by an id reused in resources.arsc

Application structure

AndroidManifest

```
<manifest package="com.example.pro" platformBuildVersionCode="34" platformBuildVersionName="14">
  <uses-permission android:name="android.permission.ACTIVITY_RECOGNITION"/>
  <application android:theme="@7F16059E" android:label="@7F150398" android:icon="@7F110000" android:debuggable="true"
    android:name="com.fitnesskeeper.runkeeper.RunkeeperApplication"
    android:networkSecurityConfig="@7F18000C" >
    <meta-data android:name="com.google.firebase.messaging.default_notification_icon" android:resource="@7F080413"/>
    <meta-data android:name="com.google.firebase.messaging.default_notification_color" android:resource="@7F06035C"/>
    <activity android:theme="@7F16052C" android:name="com.fitnesskeeper.runkeeper.SplashActivity" android:exported="true">
      <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.LAUNCHER"/>
      </intent-filter>
    </activity>
  </application>
</manifest>
```

resources.arsc

```
<public type="mipmap" name="ic_launcher" id="0x7f110000" data="res/9w.png" data_size=8/>
<public type="mipmap" name="ic_launcher" id="0x7f110000" data="res/yn.png" data_size=8/>
<public type="mipmap" name="ic_launcher" id="0x7f110000" data="res/FS.png" data_size=8/>
<public type="mipmap" name="ic_launcher" id="0x7f110000" data="res/RJ.png" data_size=8/>
<public type="mipmap" name="ic_launcher" id="0x7f110000" data="res/o-.png" data_size=8/>
<public type="mipmap" name="ic_launcher" id="0x7f110000" data="res/BW.xml" data_size=8/>
<public type="string" name="global_celsius" id="0x7f150398" data="C" data_size=8/>
<public type="style" name="TripSummary.Card.NoMapNoPhoto.Label" id="0x7f16059e" data="0x0" data_size=0/>
<public type="xml" name="network_security_config" id="0x7f18000c" data="res/8G.xml" data_size=8/>
```

Application structure

AndroidManifest

```
<manifest package="com.example.pro" platformBuildVersionCode="34" platformBuildVersionName="14">
  <uses-permission android:name="android.permission.ACTIVITY_RECOGNITION"/>
  <application android:theme="@7F16059E" android:label="@7F150398" android:icon="@7F110000" android:debuggable="true"
    android:name="com.fitnesskeeper.runkeeper.RunkeeperApplication"
    android:networkSecurityConfig="@7F18000C" >
    <meta-data android:name="com.google.firebase.messaging.default_notification_icon" android:resource="@7F080413"/>
    <meta-data android:name="com.google.firebase.messaging.default_notification_color" android:resource="@7F06035C"/>
    <activity android:theme="@7F16052C" android:name="com.fitnesskeeper.runkeeper.SplashActivity" android:exported="true">
      <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.LAUNCHER"/>
      </intent-filter>
    </activity>
  </application>
</manifest>
```

resources.arsc

```
<public type="mipmap" name="ic_launcher" id="0x7f110000" data="res/9w.png" data_size=8/>
<public type="mipmap" name="ic_launcher" id="0x7f110000" data="res/yn.png" data_size=8/>
<public type="mipmap" name="ic_launcher" id="0x7f110000" data="res/FS.png" data_size=8/>
<public type="mipmap" name="ic_launcher" id="0x7f110000" data="res/RJ.png" data_size=8/>
<public type="mipmap" name="ic_launcher" id="0x7f110000" data="res/o-.png" data_size=8/>
<public type="mipmap" name="ic_launcher" id="0x7f110000" data="res/BW.xml" data_size=8/>
<public type="string" name="global_celsius" id="0x7f150398" data="C" data_size=8/>
<public type="style" name="TripSummary.Card.NoMapNoPhoto.Label" id="0x7f16059e" data="0x0" data_size=0/>
<public type="xml" name="network_security_config" id="0x7f18000c" data="res/8G.xml" data_size=8/>
```

Application structure

AndroidManifest

```
<manifest package="com.example.pro" platformBuildVersionCode="34" platformBuildVersionName="14">
  <uses-permission android:name="android.permission.ACTIVITY_RECOGNITION"/>
  <application android:theme="@7F16059E" android:label="@7F150398" android:icon="@7F110000" android:debuggable="true"
    android:name="com.fitnesskeeper.runkeeper.RunkeeperApplication"
    android:networkSecurityConfig="@7F18000C" >
    <meta-data android:name="com.google.firebase.messaging.default_notification_icon" android:resource="@7F080413"/>
    <meta-data android:name="com.google.firebase.messaging.default_notification_color" android:resource="@7F06035C"/>
    <activity android:theme="@7F16052C" android:name="com.fitnesskeeper.runkeeper.SplashActivity" android:exported="true">
      <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.LAUNCHER"/>
      </intent-filter>
    </activity>
  </application>
</manifest>
```

resources.arsc

```
<public type="mipmap" name="ic_launcher" id="0x7f110000" data="res/9w.png" data_size=8/>
<public type="mipmap" name="ic_launcher" id="0x7f110000" data="res/yn.png" data_size=8/>
<public type="mipmap" name="ic_launcher" id="0x7f110000" data="res/FS.png" data_size=8/>
<public type="mipmap" name="ic_launcher" id="0x7f110000" data="res/RJ.png" data_size=8/>
<public type="mipmap" name="ic_launcher" id="0x7f110000" data="res/o-.png" data_size=8/>
<public type="mipmap" name="ic_launcher" id="0x7f110000" data="res/BW.xml" data_size=8/>
<public type="string" name="global_celsius" id="0x7f150398" data="C" data_size=8/>
<public type="style" name="TripSummary.Card.NoMapNoPhoto.Label" id="0x7f16059e" data="0x0" data_size=0/>
<public type="xml" name="network_security_config" id="0x7f18000c" data="res/8G.xml" data_size=8/>
```

8G.xml

```
<network-security-config>
  <debug-overrides>
    <trust-anchors>
      <certificates src="user"/>
    </trust-anchors>
  </debug-overrides>
</network-security-config>
```

Demo Reading of AXML

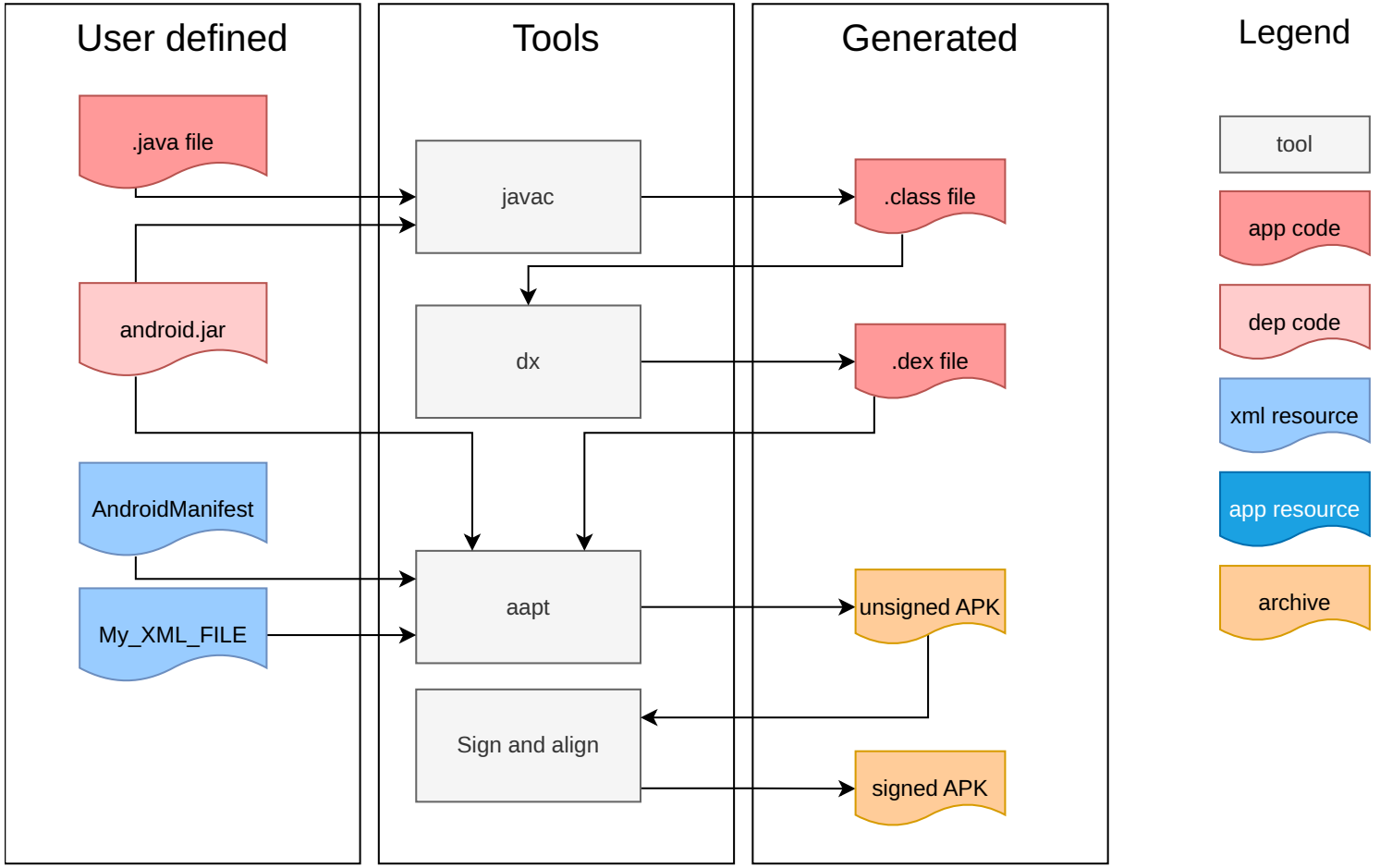
```
pip install pyaxml
```

you can have some example script here:

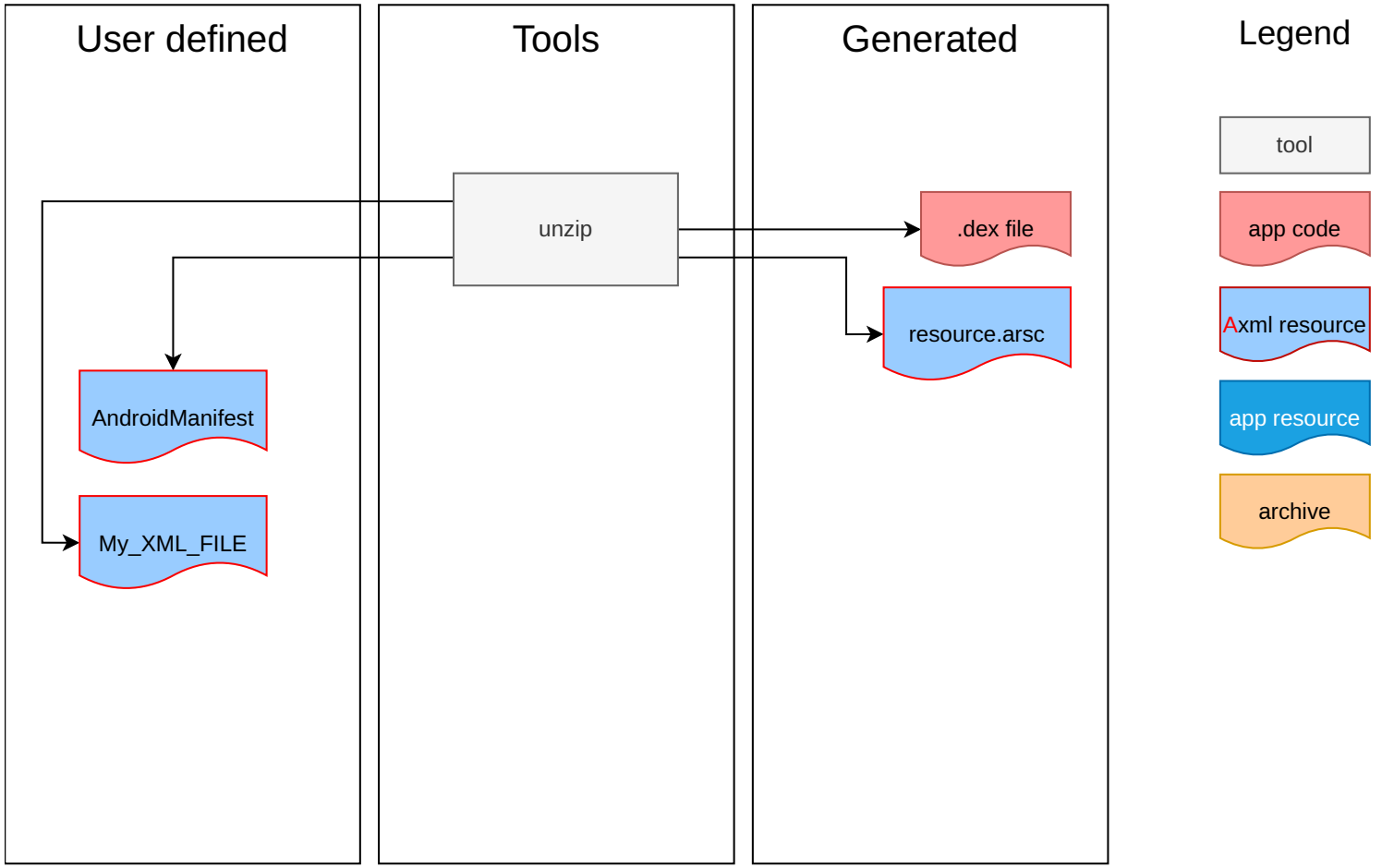
- https://gitlab.com/MadSquirrels/mobile/pyaxml/-/tree/main/examples

```
# To print Manifest in XML form
pyaxml axml2xml -i AndroidManifest.xml
# To print resources.arsc in XML form
pyaxml arsc2xml -i resources.arsc
# To convert XML to AXML
pyaxml xml2axml -i AndroidManifest.xml -o AndroidManifest.axml
```

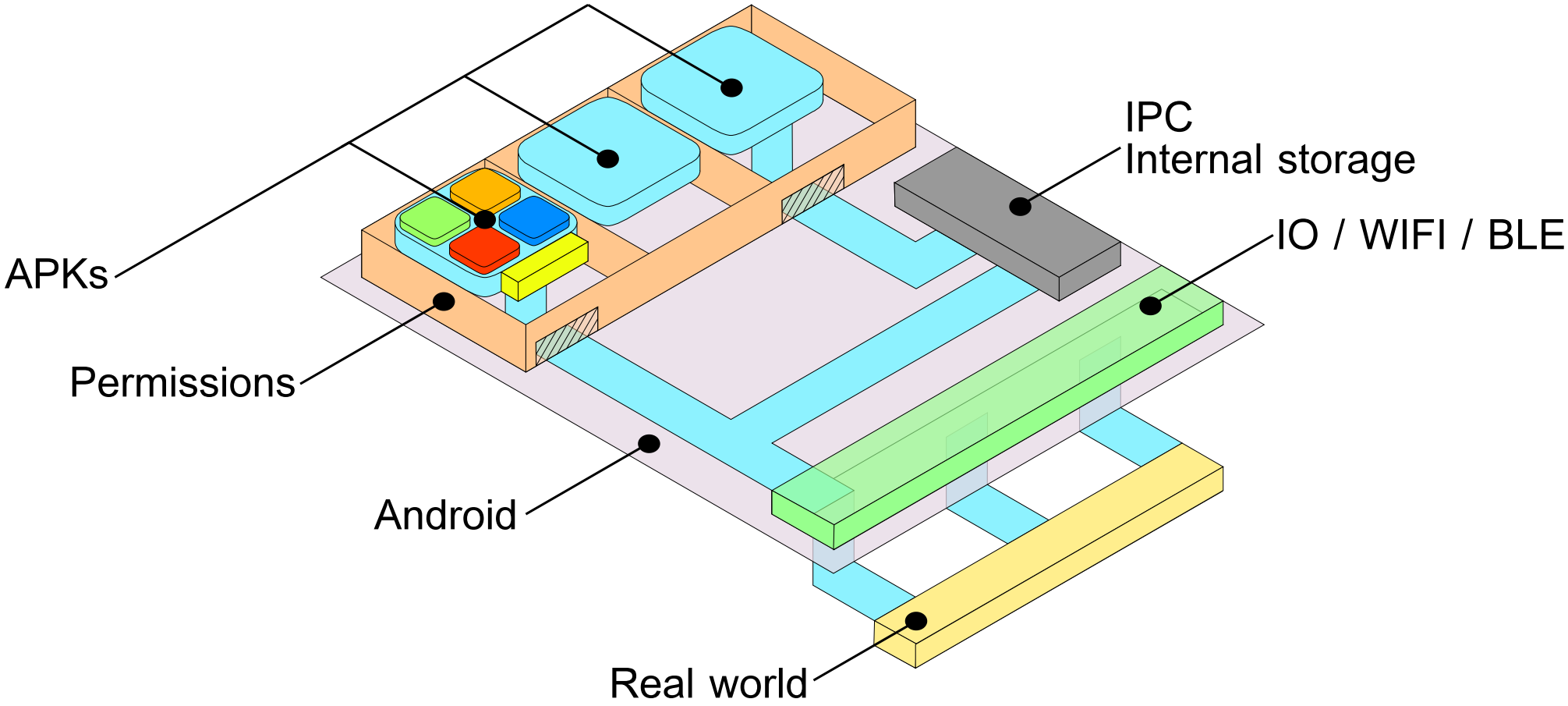
Build process



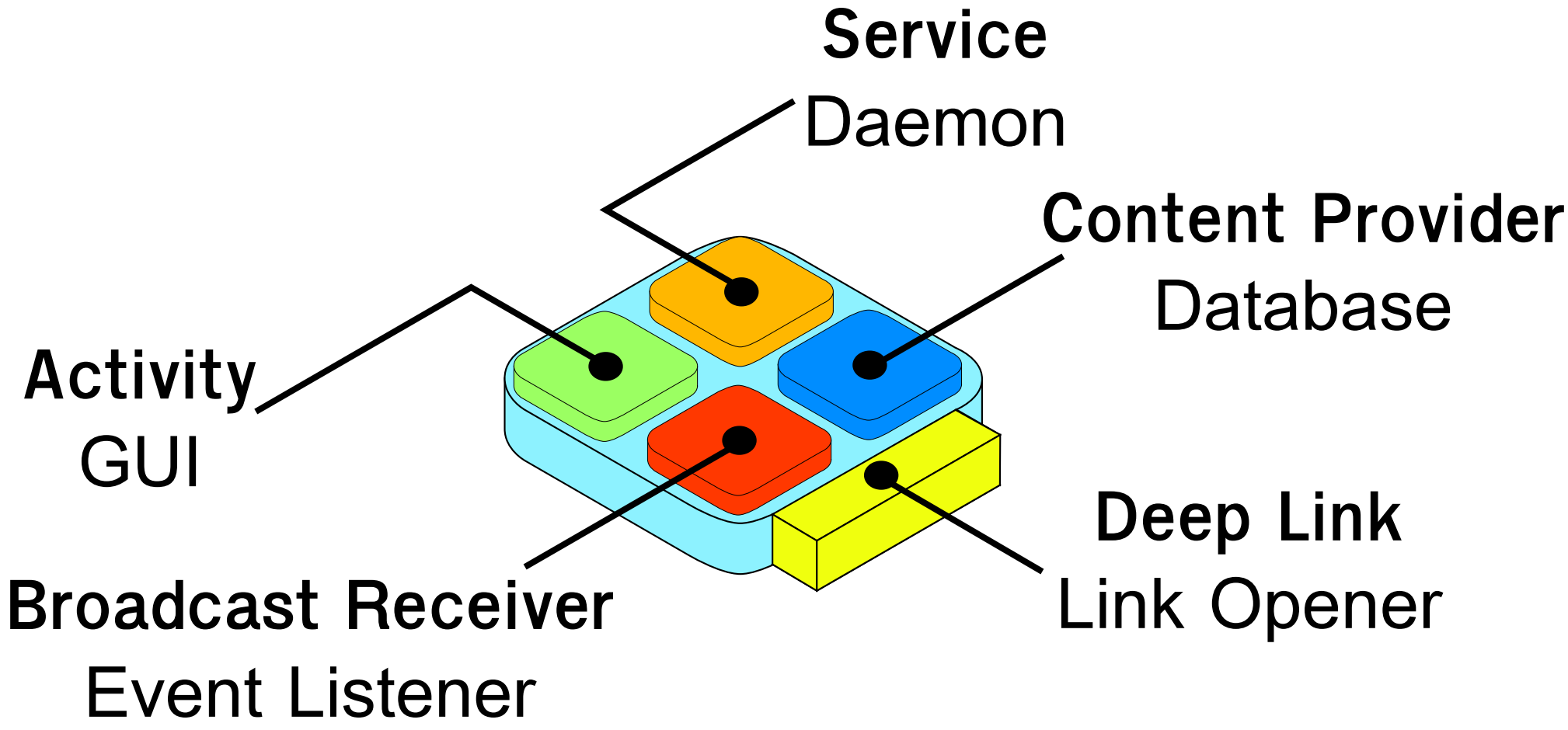
Extract process



Android application structure



Android application structure



Activity

MyActivity.java is the Declaration file.

```
import android.app.Activity;
public class MyActivity extends Activity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
  ...
  }
...
}
```

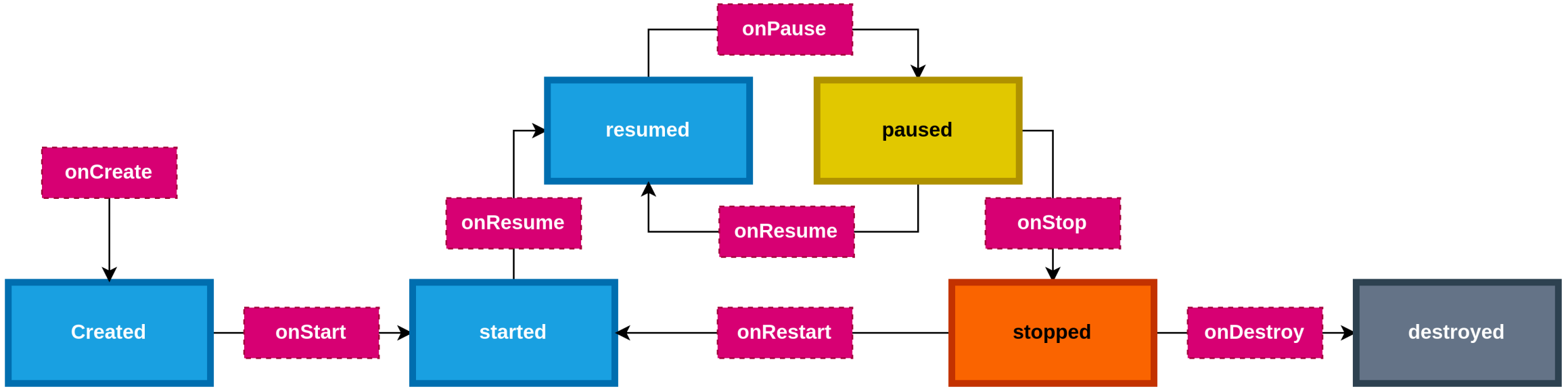
Activty

Registration of MyActivity inside AndroidManifest

```
<activity android:name="com.example.MyActivity">
<intent-filter>
<action android:name="android.intent.action.MAIN"/>
<category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
```

Activity

Lifcycle of an activity:



Service

MyActivity.java is the Declaration file.

```
public class HelloService extends Service {

  @Override
  public void onCreate() {
    ...
  }
  ...
}
```

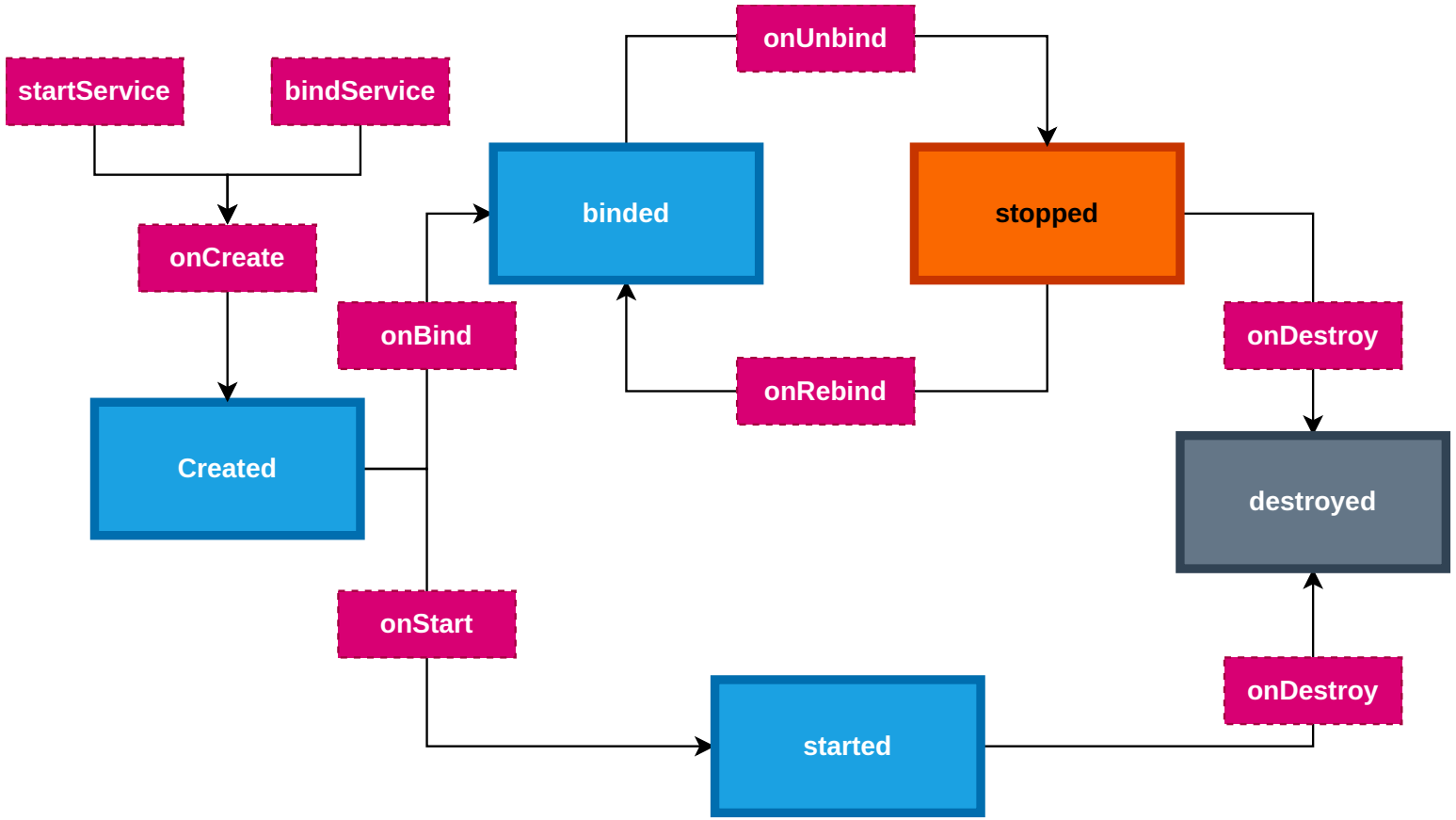
Service

Registration of MyService inside AndroidManifest

```
<manifest ... >
  ...
  <application ... >
      <service android:name="com.example.MyService" />
      ...
  </application>
</manifest>
```

Service

Lifcycle of a service:



Broadcast receiver

MyBroadcastReceiver.java is the Declaration file.

```
public static class MyBroadcastReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        if (Objects.equals(intent.getAction(), "com.example.snippets.ACTION_UPDAT
            String data = intent.getStringExtra("com.example.snippets.DATA");
            // Do something with the data, for example send it to a data reposito
            if (data != null) { dataRepository.updateData(data); }
        }
    }
}
```

Broadcast receiver

Registration of MyBroadcastReceiver inside AndroidManifest

```
<manifest ... >
  ...
  <receiver android:name=".MyBroadcastReceiver" android:exported="false">
      <intent-filter>
          <action android:name="com.example.snippets.ACTION_UPDATE_DATA" />
      </intent-filter>
  </receiver>
</manifest>
```

Content Provider

MyContentProvider.java is the Declaration file.

```
public class MyContentProvider extends ContentProvider {

    public static final String AUTHORITY = "com.example.myapp.provider";
    public static final Uri CONTENT_URI = Uri.parse("content://" + AUTHORITY + "/

    @Override
    public boolean onCreate() {
    }

    @Override
    public Cursor query(Uri uri, String[] projection, String selection,
                        String[] selectionArgs, String sortOrder) {
        ...
    }
}
```

Content Provider

Registration of MyContentProvider inside AndroidManifest

```
<manifest ... >
  ...
  <provider
              android:name=".MyContentProvider"
              android:authorities="com.example.myapp.provider"
              android:exported="true"
              android:grantUriPermissions="true" />
</manifest>
```

Deeplink

DeepLinkActivity.java is the Declaration file.

```
public class DeepLinkActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        Uri data = getIntent().getData();
        if (data != null) {
            // Handle the deep link here
        }

        setContentView(R.layout.activity_deep_link);
    }
}
```

Deeplink

Registration of DeepLinkActivity inside AndroidManifest

```
<manifest ... >
  ...
  <activity android:name=".DeepLinkActivity">
    <intent-filter>
      <action android:name="android.intent.action.VIEW" />
      <category android:name="android.intent.category.DEFAULT" />
      <category android:name="android.intent.category.BROWSABLE" />
      <data
        android:scheme="https"
        android:host="www.example.com"
        android:pathPrefix="/open" />
    </intent-filter>
  </activity>
</manifest>
```

How to use Apkpatcher

A quick way to use it:

```
docker run --rm -v .:/pwd -it madsquirrels/apkpatcher -a base.apk
```

The hard way

```
# Java dependencies
apt install -y default-jre

# sdktools dependendies installation
wget https://dl.google.com/android/repository/commandlinetools-linux-6200805_late
mkdir /usr/lib/android-sdk
cd /usr/lib/android-sdk
unzip commandlinetools-linux-6200805_latest.zip
mkdir cmdline-tools
mv tools/ cmdline-tools/
echo 'export ANDROID_SDK_ROOT=/usr/lib/android-sdk' >> ~/.bashrc

# installation of platform-tools
sdkmanager "platform-tools" "platforms;android-36" "build-tools;36.0.0" "emulator

# install apkpatcher
pip install apkpatcher
```

Some example of usage

```
apkpatcher -a <apk> -m <split_apk1> <split_apk2>
```

To enable debug mode

```
apkpatcher -a <apk>  --enable-debug
```

If you want to begin to debug you follow this tutorial or use for instance jadx:

- https://apkpatcher.ci-yow.com/how.html#enabling-debuggable-mode

To inject frida or a library

To do it simply:

```
apkpatcher -a <apk> --download_frida_version <version>
```

To choose a specific library:

```
apkpatcher -a <apk> -g <file.so> -r arm64
```

To change location which inject:

```
apkpatcher -a <apk> -g <file.so> -r arm64 --entrypoint com.example.entrypoint
```

Demo frida

installation:

```
$ pip install frida-tools
$ frida --version
16.7.13
$ apkpatcher -a <apk> --download_frida_version 16.7.13
```

Documentation for frida here:

- https://frida.re/docs/examples/android/

To add a permission

```
apkpatcher -a <apk> --add-permissions <my_permission>
```

Add a plugin or set a pause during the process

To pause the process:

```
apkpatcher -a <apk> -p
```

To execute a script during the process:

```
apkpatcher -a <apk> --plugin
```

Little Smali introduction

Java code:

```
public static int compare(long j, long j2) {
    if (j < j2) {return -1;}
    return j == j2 ? 0 : 1;
}
```

Smali code:

```
.method public static compare(JJ)I
    .registers 4
    cmp-long            v0, v0, v2
    ...
    return              v0
.end method
```

Little Smali introduction

- Dalvik Intermediate Representation is called Smali: It can be seen as an assembly language that is human readable and can be used to generate Dalvik bytecode.
- Primitive types are simply mangled with a letter.
- Object types are mangled as follows: Ljava/lang/String;

Little Smali introduction

Native types are the following:

- V void, which can only be used for return value types
- Z boolean
- B byte
- S short
- C char
- I int
- J long (64 bit)
- F float
- D double (64 bit)

Little Smali introduction

Call to methods:

Class definition in JAVA

```
package challenge.examples;
class MyClass {
  bool method();
}
```

Call in Smali

```
Lchallenge/examples/MyClass;->method()Z
```

Little Smali introduction

more information on smali here: https://blog.quarkslab.com/smali-the-parseltongue-language.html

Patch the smali code

```
from apkpatcher.smalipatching import Method, get_smali_file_from_class, replace_m
import click


def replace_method_wrapper(input_dir, classname, prototype, patch):
    m = Method(prototype, patch)
    fname = get_smali_file_from_class(input_dir, classname)
    with open(f"{fname}", "r") as f:
        content = f.read()
    with open(f"{fname}", "w") as f:
        text = replace_methods([m], content)
        print(text)
        content = f.write(text)
```

Patch the smali code

```
@click.command()
@click.argument('input_dir', nargs=-1)
def main(input_dir):
    main_dir=input_dir[0]
    prototype = ".method public setEndColor(I)V"
    patch = "return-void"
    classname = "com.github.mikephil.charting.model.GradientColor"
    replace_method_wrapper(main_dir, classname, prototype, patch)
```

Demo patching

Inject a burp certificate

Accept user certificate:

```
apkpatcher -a <apk> -e
```

Add a custom certificate:

```
apkpatcher -a <apk> -c burp.der
```

Just a little reminder about ADB and enable debug mode

Enable ADB

On the device, go to Settings > About .
Tap the Build number seven times to make Settings > Developer options available.
Then enable the USB Debugging option.

Some little ADB command

```
adb install <apk>
adb shell
adb logcat
```

Link for challenge

https://transfer.ci-yow.com/d?id=zPsd93uWjSmjALn

Thank you!

Contact information:

apkpatcher: https://apkpatcher.ci-yow.com

Email: bforgette@quarkslab.com

Twitter: https://twitter.com/Mad5quirrel

Quarkslab