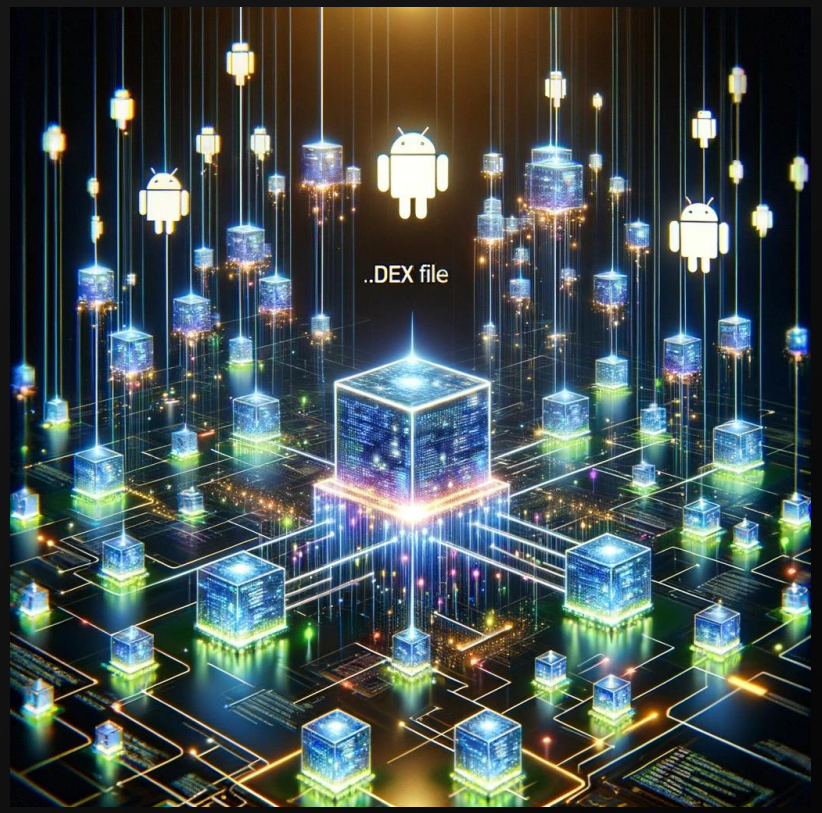
Module 9 – Execute Custom .dex File

It’s dangerous to load .dex files…

MALICIOUS ERECTION LLC

Executing Custom `.dex` File

- As you should know, Android code gets compiled into a bunch of `classes.dex` files within an `.apk` file
- Application developers have the ability to dynamically load arbitrary `.dex` files into their running application, essentially adding code on the fly
  - Malware developers tend to use this feature so that their malware gets dynamically added to “legitimate applications”
- This module will focus on this `.dex` file loading feature and how to take advantage of it



DALL-E created image describing what a `.dex` file is

MALICIOUS ERECTION LLC

Executing Custom `.dex` File

- Some documentation does exist about this feature
  - 2011 article - https://android-developers.googleblog.com/2011/07/custom-class-loading-in-dalvik.html
- There is also user generated content and blogs, which may be more useful
  - https://kalpeshchandora12.medium.com/dynamic-code-loading-in-android-dea83ba3bc85
  - https://medium.com/@artyomdangizyan/aar-to-dex-loading-and-running-code-at-runtime-in-android-application-69089a30c715
  - https://erev0s.com/blog/3-ways-for-dynamic-code-loading-in-android/
- All of these articles give different recommendations on how to:
  - Craft a `.dex` file
  - Load a `.dex` file
  - Executing a `.dex` file
- If this feature is being used by a legitimate non-malware application, then ideally the `.dex` file is loaded from a secure location
  - Such as the application's Assets folder or a remote server controlled by the developer
- Next, lets observe how Axolotl loads `.dex` files

MALICIOUS ERECTION LLC

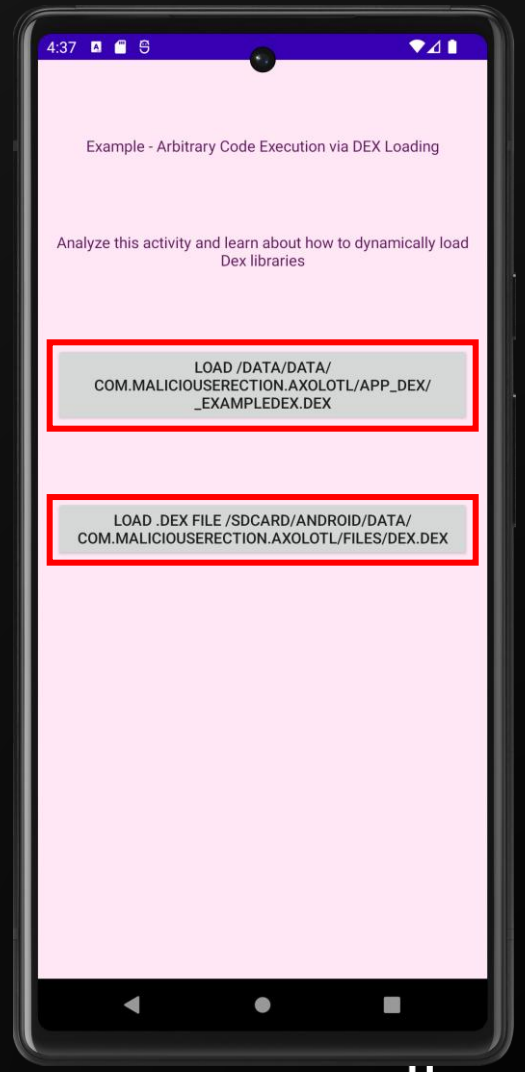
Executing Custom `.dex` File - Example

- We will now use Axolotl to better demonstrate how to execute custom `.dex` files
- On Axolotl's main menu, tap:
  - "Exercise Modules"
  - "Execute Custom .dex File"
- A blank activity will appear with some text
- The launched activity is programmed via the Java class `com.maliciouserection.axolotl.example.activity.codeExecution.dexClassLoader`

MALICIOUS ERECTION LLC

Executing Custom .dex File - Example

- There are two buttons on the `dexClassLoader` Activity
  - The first button loads the file `/data/data/com.maliciouserection.axolotl/app_dex/_exampleDex.dex` and executes it
  - The second button loads the file `/sdcard/Android/data/com.maliciouserection.axolotl/files/dex.dex` and executes it
- Decompiling the Activity reveals that Axolotl copies `_exampleDex.dex` from the Assets folder to `/data/data/com.maliciouserection.axolotl/app_dex`




```
public class dexClassLoader extends Activity {
    public static String dexFile = "_exampleDex.dex";
    TextView text;
```

```
place_flags_and_files.copyFromAssetsToInternalDex(getApplicationContext(), dexFile);
executeDex(dexFile);
```

Decompiled code for `dexClassLoader`

MALICIOUS ERECTION LLC

Executing Custom .dex File - Example

- A quick note about the `app_dex` folder
- In Android, if you use the `getDir(String, int)` API to retrieve an internal directory, and you specify the folder `dex` as the String argument, then Android automatically converts `dex` into `app_dex`
- This is why Axolotl's source code contains the code `getDir("dex", 0)` with `dex` being the String argument

```
public static void copyFromAssetsToInternalDex(Context context, String str) {
    try {
        FileOutputStream fileOutputStream = new FileOutputStream(new File(context.getDir("dex", 0), str));
        InputStream open = context.getAssets().open(str);
        byte[] bArr = new byte[1024];
        while (true) {
            int read = open.read(bArr);
            if (read > 0) {
                fileOutputStream.write(bArr, 0, read);
            } else {
                fileOutputStream.close();
                return;
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Example usages of `getDir("dex", 0)` in Axolotl

```
private void executeDex(String dexFile2) {
    File yayDexStoragePathYay = new File(getDir("dex", 0), dexFile2);
    DexClassLoader yayClassLoaderYay = new DexClassLoader(yayDexStoragePathYay.getAbsolutePath(), null, null, getClassLoader());
    try {
        Class<?> theClass = yayClassLoaderYay.loadClass("com.yogehi.exampledex.yayclassyay");
        Method theMethod = theClass.getMethod("yaymethodyay", Context.class, String.class);
        theMethod.invoke(theClass.newInstance(), getApplicationContext(), "You sure axolotl questions");
```

MALICIOUS ERECTION LLC

Executing Custom `.dex` File - Example

- Looking at the source code for the first button, tapping it takes the value `` `_exampleDex.dex` `` and passes it as an argument to the method `` `executeDex(String)` ``
- `` `executeDex(String)` `` will perform the following actions:
  - Load the specified `.dex` file from `` `/data/data/com.maliciouserection.axolotl/app_dex/` ``
  - From the `.dex` file, search and load the class `` `com.yogehi.exampledex.yayclassyay` ``
  - From the loaded class, search for the method `` `yaymethodyay(Context, String)` ``
  - Execute that method with the arguments `` `getApplicationContext()` `` and "You sure axolotl questions"

```
this.yaybuttonyay1.setOnClickListener(new View.OnClickListener() {
    @Override // android.view.View.OnClickListener
    public final void onClick(View view) {
        dexClassLoader.this.m106x61a5ac73(view);
    }
```

```
public /* synthetic */ void m106x61a5ac73(View v) {
    executeDex(dexFile);
}
```

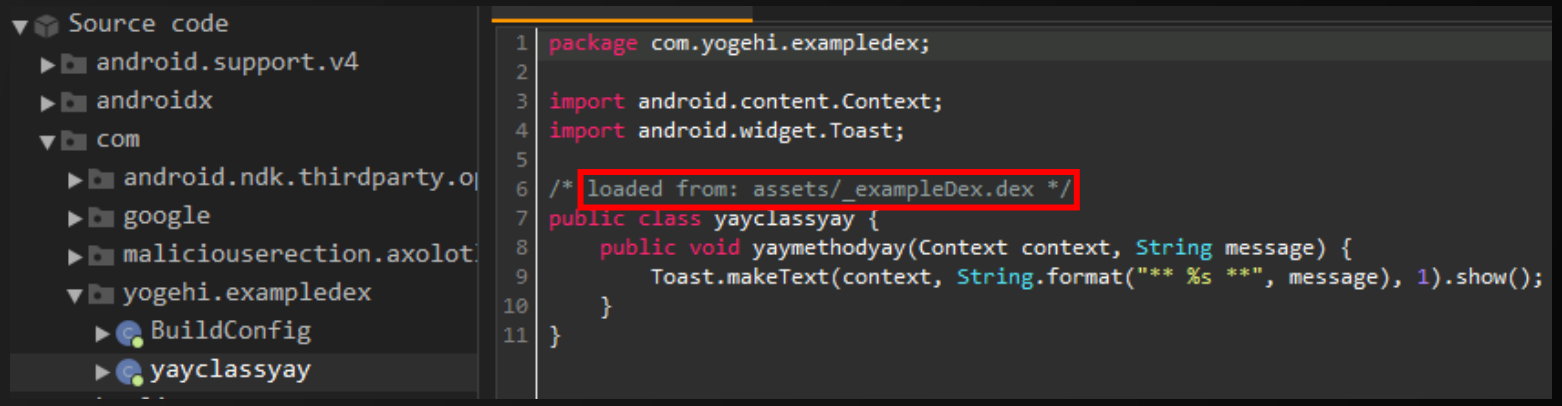
```
private void executeDex(String dexFile2) {
    File yayDexStoragePathYay = new File(getDir("dex", 0), dexFile2);
    DexClassLoader yayClassLoaderYay = new DexClassLoader(yayDexStoragePathYay.getAbsolutePath(), null, null, getClassLoader());
    try {
        Class<?> theClass = yayClassLoaderYay.loadClass("com.yogehi.exampledex.yayclassyay");
        Method theMethod = theClass.getMethod("yaymethodyay", Context.class, String.class);
        theMethod.invoke(theClass.newInstance(), getApplicationContext(), "You sure axolotl questions");
```

Decompiled code for `` `dexClassLoader` ``

MALICIOUS ERECTION LLC

Executing Custom `.dex` File - Example

- So at this point, we want to view the source code of `com.yogehi.exampledex.yayclassyay`
  - Based on the decompiled Axolotl code, this class should be in the `.dex` file `_exampleDex.dex`
- We have some options to view the source code for that `.dex` file
  - Put the entire Axolotl `.apk` file into JADX, which will automatically detect the `.dex` file in the Assets folder and decompile it
  - Change the extension of the `.apk` file to `.zip`, unzip the archive, extract the `.dex` file from the Assets folder, and decompile the `.dex` file via your preferred method

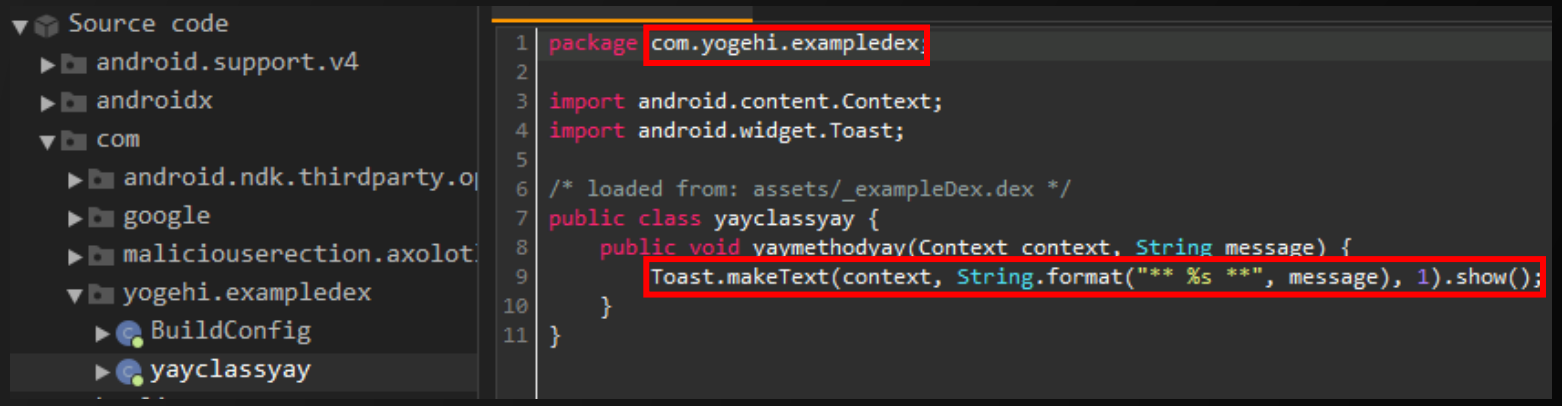


Decompiled code for `_exampleDex.dex` taken from Axolotl's Assets folder

MALICIOUS ERECTION LLC

Executing Custom `.dex` File - Example

- Looking at the source code for `yayclassyay`, we can see that the package name is `com.yogehi.exampledex`
- We can also see that the method `yaymethodyay` creates a Toast message
- The contents of the Toast message will be a string formatted String, with a part of the message being taken from one of the arguments used to execute `yaymethodyay`



Decompiled code for `_exampleDex.dex` taken from Axolotl's Assets folder

MALICIOUS ERECTION LLC

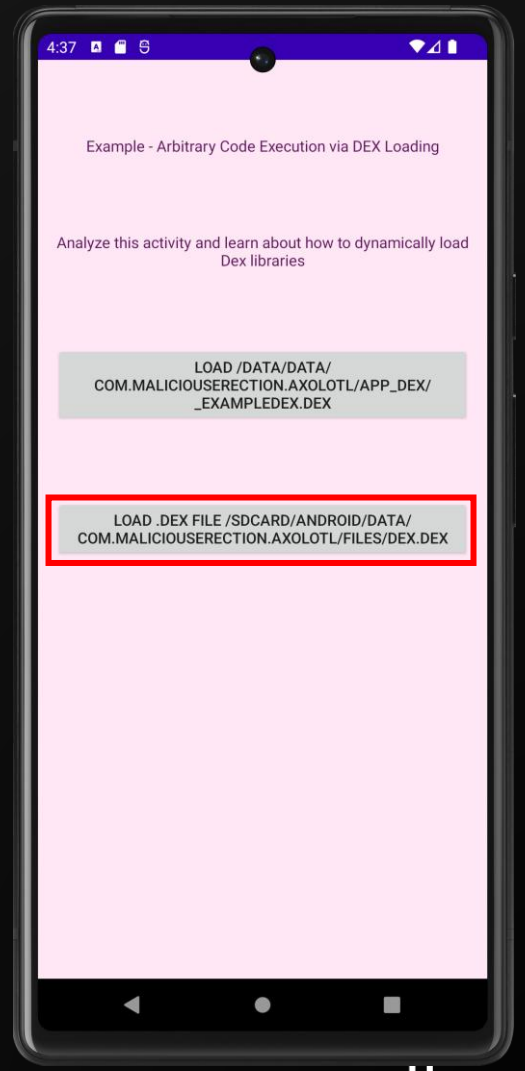
Executing Custom `.dex` File - Example

- So, to summarize, at a high level, tapping the first button:
  - Load the specified `.dex` file from `` `/data/data/com.maliciouserection.axolotl/app_dex/` ``
  - From the `.dex` file, search and load the class `` `com.yogehi.exampledex.yayclassyay` ``
  - From the loaded class, search for the method `` `yaymethodyay(Context, String)` ``
  - The loaded method is executed with the arguments `` `getApplicationContext()` `` and “You sure axolotl questions”
  - The executed method will create a Toast message with the contents “** <String argument> **”
    - So the Toast contents will be “** You sure axolotl questions **”
- After tapping the first button, you should see the Toast message appear
  - The Toast message appearing confirms the above summary

MALICIOUS ERECTION LLC

Executing Custom `.dex` File - Example

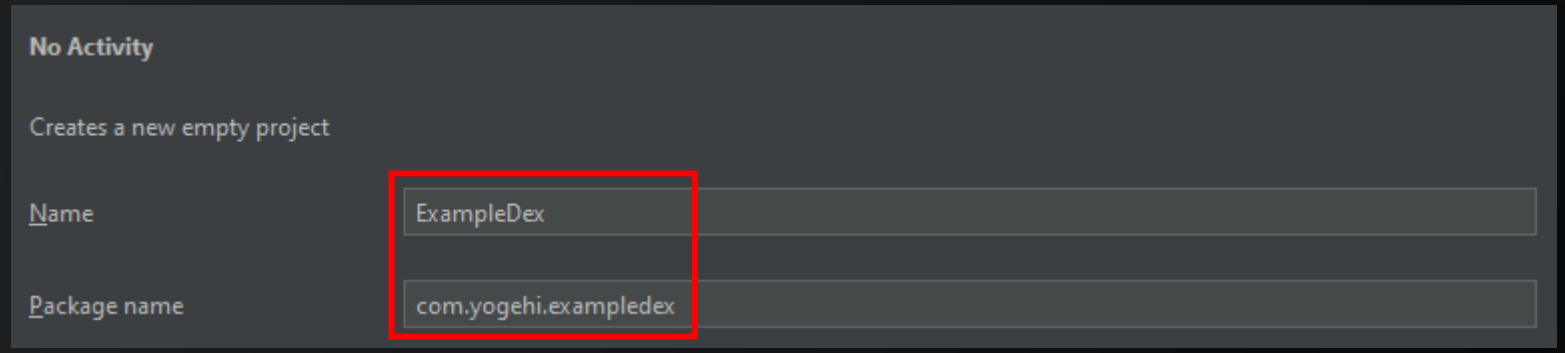
- We know what the first button does and how Axolotl executes a `.dex` file
- What does the second button do?
- As previously stated, the second button loads the file `` `/sdcard/Android/data/com.maliciouserection.axolotl/files/dex.dex` `` and executes it
  - If you decompile and analyze `dexClassLoader`, you should see how Axolotl will copy `dex.dex` from `/sdcard/Android/data/com.maliciouserection.axolotl/files/` to `/data/data/com.maliciouserection.axolotl/app_dex/`
  - Then, `executeDex(String)` is executed with the `dex.dex` file as an argument
- So, lets try to create the file `dex.dex` and place it in the appropriate folder for Axolotl to execute



MALICIOUS ERECTION LLC

Executing Custom `.dex` File - Example

- We will be using Android Studio to make the `dex.dex` file
- Open Android Studio and make a new project without an Activity
- Name the application “ExampleDex” and the package name `com.yogehi.exampledex`
  - We are using that package name since that is the one used in `_exampleDex.dex`



Android Studio window while creating the project

Executing Custom .dex File - Example

- After the project is created, add the class `yayclassyay`
- Then, add the method `yaymethodyay` with the same arguments that Axolotl executes
  - If you recall, `executeDex(String)` executes `yaymethodyay(Context, String)`
  - So, the arguments should be `Context` and `String`

```
private void executeDex(String dexFile2) {
    File yayDexStoragePathYay = new File(getDir("dex", 0), dexFile2);
    DexClassLoader yayClassLoaderYay = new DexClassLoader(yayDexStoragePathYay.getAbsolutePath(), null, null, getClassLoader());
    try {
        Class<?> theClass = yayClassLoaderYay.loadClass("com.yogehi.exampledex.yayclassyay");
        Method theMethod = theClass.getMethod("yaymethodyay", Context.class, String.class);
        theMethod.invoke(theClass.newInstance(), getApplicationContext(), "You sure axolotl questions");
```

(Left) Axolotl's `executeDex(String)` which executes `yaymethodyay(Context, String)`

```
package com.yogehi.exampledex;

import android.content.Context;

no usages
public class yayclassyay {

    no usages
    public void yaymethodyay(Context context, String string) {
        // code will go here
    }
}
```

(Right) Source code for ExampleDex's `yayclassyay` which contains `yaymethodyay(Context, String)`

Executing Custom .dex File - Example

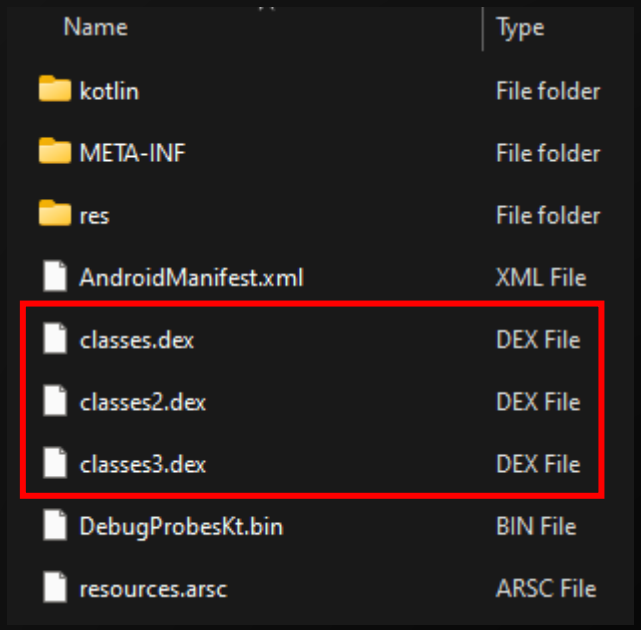
- Let's add some code to our `yaymethodyay(Context, String)`
  - `Log.i("yaylogyay", "executed an arbitrary dex file");`
    - This code will add an `adb` log entry with the tag `yaylogyay`
  - `Toast.makeText(context, "executed an arbitrary dex file", Toast.LENGTH_LONG).show();`
    - This code will create a Toast message

```
package com.yogehi.exampledex;

import android.content.Context;
import android.util.Log;
import android.widget.Toast;

no usages
public class yayclassyay {

    no usages
    public void yaymethodyay(Context context, String string) {
        Log.i( tag: "yaylogyay",  msg: "executed an arbitrary dex file");
        Toast.makeText(context,  text: "executed an arbitrary dex file", Toast.LENGTH_LONG).show()
    }
}
```

Source code for `yayclassyay`

Executing Custom `.dex` File - Example

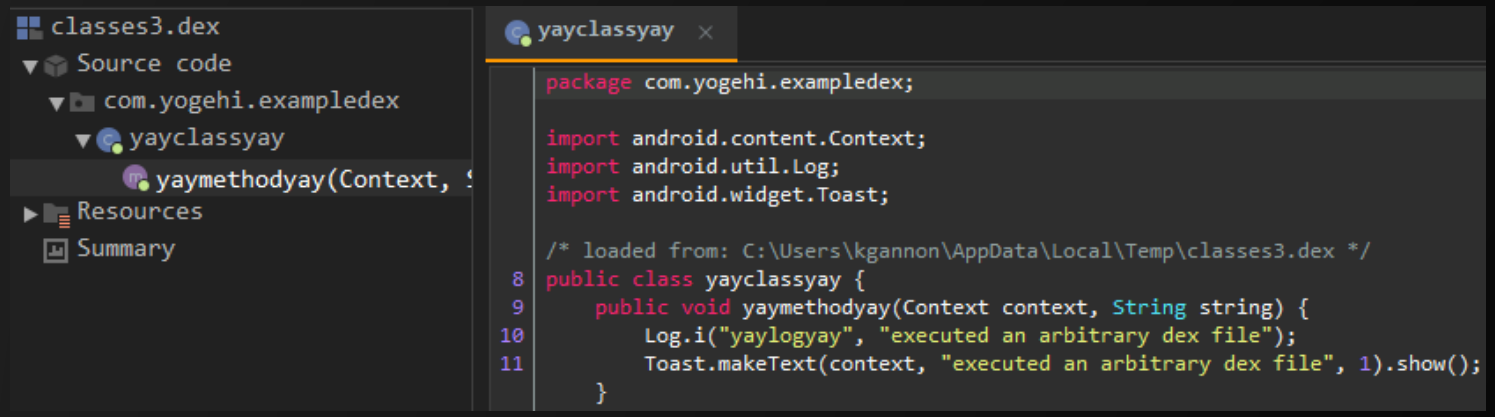
- Next, build an `.apk` file
  - Top bar -> Build -> Build Bundle(s) / APK(s) -> Build APK(s)
- Locate the `.apk` file on your host and change the extension of the file from `.apk` to `.zip`
- Open the `.zip` file and there should be some `.dex` files
  - NOTE: the number of `.dex` files you have may differ from the screenshot below



MALICIOUS ERECTION LLC

Executing Custom `.dex` File - Example

- If you put each `.dex` file into JADX, you can view which `.dex` file contains your newly created `` `yayclassyay` ``
  - In this case, `` `yayclassyay` `` was located in `` `classes3.dex` ``
- Take a note of which `.dex` file contained `` `yayclassyay` `` and put that `.dex` file in a convenient location on your host



JADX revealing which `.dex` file contained `` `yayclassyay` ``

MALICIOUS ERECTION LLC

Executing Custom `.dex` File - Example

- Rename `classes3.dex` to `dex.dex`
- Push `dex.dex` to `/sdcard/Android/data/com.maliciouserection.axolotl/files/`
- Open the `dexClassLoader` Activity again and tap the second button
- You should see your Toast and Log messages appear
  - This confirms that your arbitrary `.dex` file was executed

```
c:\>adb push dex.dex /sdcard/Android/data/com.maliciouserection.axolotl/files/
dex.dex: 1 file pushed, 0 skipped. 0.1 MB/s (1012 bytes in 0.008s)

c:\>adb shell
emulator64_x86_64_arm64:/ $ logcat | grep yaylogyay
01-15 22:53:05.844  7950  7950 I yaylogyay: executed an arbitrary dex file
```

MALICIOUS ERECTION LLC

Module 9 Exercise

- Capture The Flag - There is a class `com.maliciouserection.axolotl.util.theDexFlag` which does not get imported or used anywhere else in Axolotl
- In that class is the method `logDexFlag()` which will output Flag 9 in Logcat
- Use the Activity `dexClassLoader` to execute `logLastFlag()` and get the last flag.
  - This exercise does not require Example Exploit; you can finish this exercise by physically touching the device
  - Executing arbitrary `.dex` files gives you RCE over the vulnerable application
  - So the flag for this exercise is not as well hidden as the other flags

```
emulator64_x86_64_arm64:/ $ logcat | grep axolotl_flag
01-15 23:00:44.564  7950  7950 I axolotl_flag: Flag 9: [redacted]
```

MALICIOUS ERECTION LLC