Graham Dungan
February 18, 2025
UIN: 332001764

CSCE 413: Software Security
Class 14: APKs

Demo

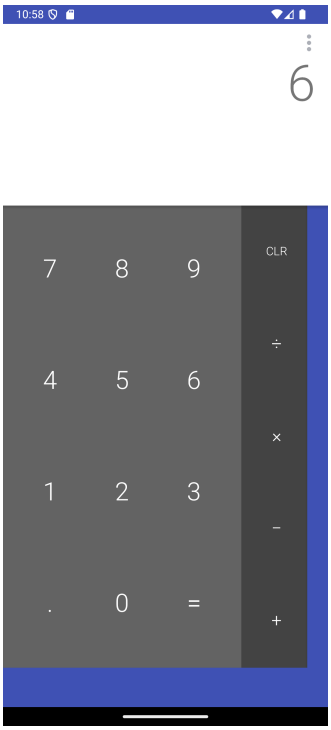
A video demo of all material presented within this writeup is available at: https://youtu.be/NDOIkZIr6QM
Alongside this document are two files; `calculator.apk`, the original unmodified APK, and `calculator_modified.apk`, the modified APK with extra functionality.

Choosing an APK

For this assignment, I chose to add extra functionality to a simple calculator app, "Calculator" by FTA Apps. I acquired the Calculator APK from ApkPure. When looking for an APK, I specifically looked for simple, somewhat old, unpopular apps. These attributes increase the possibility that the app is not too complex and does not have recent support, making it easier to understand and modify smali code.

Baseline Functionality

The Calculator app behaves as any calculator would. The screenshot provided demonstrates the outcome of adding 1+2+3, and pressing the "=", or equals, operation.



For this assignment, we will be adding extra functionality to the equals operation. Many malicious apps use scareware or squatting to convince a user that they need to update their software by downloading a new version of the app. Because of this, we will be adding a website redirect every time the user performs the equals operation. For the purposes of this assignment, the app will redirect to a static image of the famous streamer Jerma985.

Modifying the APK

Decompiling

In order to modify the APK, we will first need to decompile it. Using Android Studio's apktool, we will decompile `calculator.apk` into the directory `calculator_decompiled`.

```
PS C:\Users\user\Desktop\RepackageAtk>
>> apktool d calculator.apk -o calculator_decompiled
I: Using Apktool 2.11.0 on calculator.apk with 8 threads
I: Baksmaling classes.dex...
I: Loading resource table...
I: Baksmaling classes3.dex...
I: Baksmaling classes2.dex...
I: Decoding file-resources...
I: Loading resource table from file: C:\Users\user\AppData\Local\apktool\framework\1.apk
I: Decoding values */* XMLs...
I: Decoding AndroidManifest.xml with resources...
I: Regular manifest package...
I: Copying original files...
I: Copying assets...
I: Copying kotlin...
I: Copying META-INF/services...
I: Copying unknown files...
Press any key to continue . . .
PS C:\Users\user\Desktop\RepackageAtk>
```

Smali Code

The code that we will be modifying is found in `BasicCalculator.smali` and is within the `onEqual()` method. Since we are not removing any functionality from the app, we will not remove any code, and instead will add the following lines to the beginning of the function,

```
1  .method protected onEquals()V
2      .locals 6
3      const-string v3, "android.intent.action.VIEW"
4      const-string v4, "https://static.wikia.nocookie.net/youtube/images/d/da/Jerma985
           -2018.jpg/revision/latest?cb=20210112042437"
5      invoke-static {v4}, Landroid/net/Uri;->parse(Ljava/lang/String;)Landroid/net/Uri;
6      move-result-object v4
7      new-instance v5, Landroid/content/Intent;
8      invoke-direct {v5, v3, v4}, Landroid/content/Intent;-><init>(Ljava/lang/String;
           Landroid/net/Uri;)V
9      invoke-virtual {p0, v5}, Landroid/app/Activity;->startActivity(Landroid/content/
           Intent;)V
```

Lines 2-4 establish the number of registers needed for the function, a string declaring the type of intent we want to use, and a string with the URL to our website. Lines 5-6 create a URI from the provided URL and store it in the v4 register. Lines 7-8 use the URI, the intent, and the new instance of an Intent in register v5 to initialize an intent with the URL/URI. Line 9 invokes an activity, this activity being any activity that can handle the VIEW intent (which is the browser). This is the final step in launching the website to be viewed.

Recompiling

Now that the APK has been modified, we will need to recompile it, similarly using apktool. We will compile the new APK to the file `calculator_modified.apk`.

```
PS C:\Users\user\Desktop\RepackageAtk>
>> apktool b calculator_decompiled -o calculator_modified.apk
I: Using Apktool 2.11.0 on calculator_modified.apk with 8 threads
I: Checking whether sources have changed...
I: Checking whether sources have changed...
I: Smaling smali folder into classes.dex...
I: Smaling smali_classes2 folder into classes2.dex...
I: Checking whether sources have changed...
I: Smaling smali_classes3 folder into classes3.dex...
I: Checking whether resources have changed...
I: Building resources with aapt2...
I: Building apk file...
I: Importing assets...
I: Importing kotlin...
I: Importing META-INF/services...
I: Importing unknown files...
I: Built apk into: calculator_modified.apk
Press any key to continue . . .
PS C:\Users\user\Desktop\RepackageAtk>
```

Keygen and Signing

Android requires APKs to be signed before they are installed. Because of this, we will use `keygen` and `apksigner` to do so. The following images demonstrate the generation of a simple keystore and signing `calculator_modified.apk`.

```
PS C:\Users\user\Desktop\RepackageAtk>
>> keytool -genkey -v -keystore key.keystore -alias alias -keyalg RSA
Enter keystore password:
Re-enter new password:
Enter the distinguished name. Provide a single dot (.) to leave a sub-component empty or press ENTER to use the default value in braces.
What is your first and last name?
What is the name of your organizational unit?
What is the name of your organization?
What is the name of your City or Locality?
What is the name of your State or Province?
What is the two-letter country code for this unit?
Is CN=Unknown, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown correct?
  [no]:  y
Generating 3,072 bit RSA key pair and self-signed certificate (SHA384withRSA) with a validity of 90 days
        for: CN=Unknown, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown
[Storing key.keystore]
PS C:\Users\user\Desktop\RepackageAtk>
```

```
PS C:\Users\user\Desktop\RepackageAtk>
>> apksigner sign --ks key.keystore --ks-key-alias alias calculator_modified.apk
Keystore password for signer #1:
PS C:\Users\user\Desktop\RepackageAtk>
```

Extra Functionality

We can now view the modified APK by installing it and performing an operation. Once the equals button is clicked, the Chrome web browser will open with our custom website. For a more thorough explanation, please view the demo referenced at the beginning of this document. What follows is a screenshot demonstrating the website being loaded.