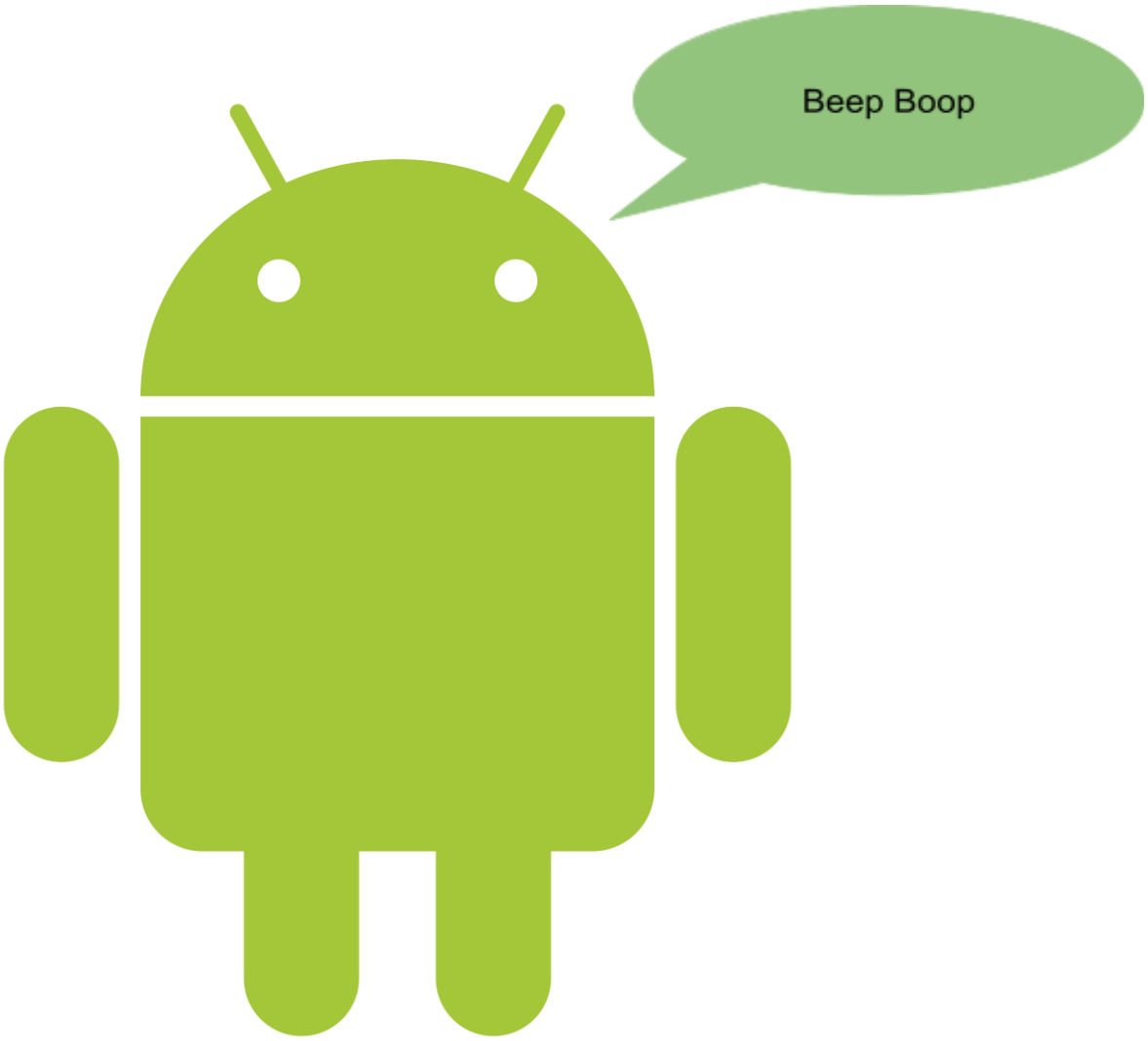
CSEC 467 - 01: Mobile Dev Security & Forensics

Lab #1: Getting Familiar with Android Application Structure

By: Mark Johnson



Critical thinking Questions:

1. In your own words, provide a short summary (about one or two paragraphs) of this lab assignment. This must describe how the activities performed in the lab address the learning outcome.

   In this lab, a basic understanding of the structure and processes of an android app was established using hands-on learning. This was accomplished by having the student create a basic android app that took in user input, changed the interface, and repeated back the inputted text.

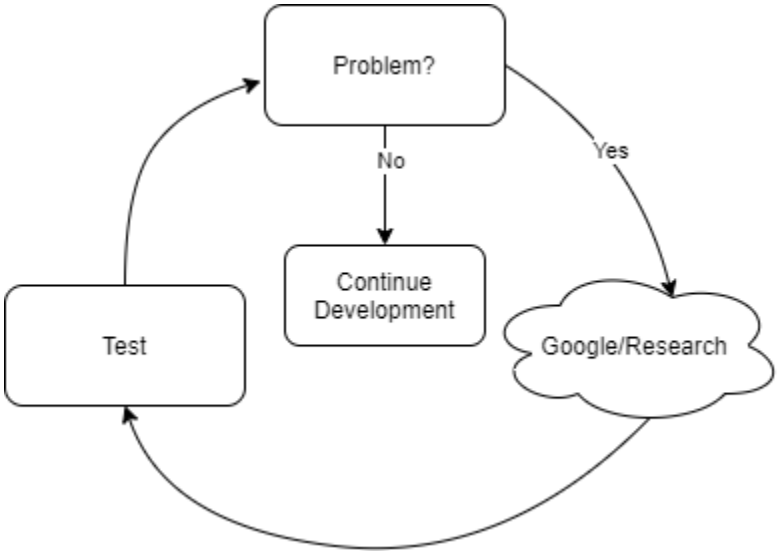
   This addressed the learning outcome by giving us the chance to see step by step, what each component is doing during the apps function. It also gave us the chance to examine the internal file structure of an android OS by creating a solitary app that we could examine from the command line. Between these two factors, a basic understanding of the android OS structure and operation was imparted.

2. Describe, in your own words, how you were able to solve this problem.

   Given that I ended up working ahead, I largely solved this problem of creating an app by googling how to do certain pieces. Initially, I went by intuition. Android Studio has a similar interface for creating GUIs as visual studio which I had worked with before, so some of the aspects of creating a user interface and setting attributes was already familiar.

   However, where I ended up googling things was when it came to creating an onClickListener. There I researched how there were several methods of doing so and I picked the one that seemed best suited to the purposes of this simple app. After I had set up the listener, I looked up how to create a new activity as well as what an activity really was. Doing so allowed me to create a new activity and create an intent to pass the information supplied by the user as well as spark the execution of the new activity. Finally, my last problem was in determining how to create a toast message. This ended up being the easiest problem as it was a simple one line solution.

3. Create a visual representation of your solution to this problem.

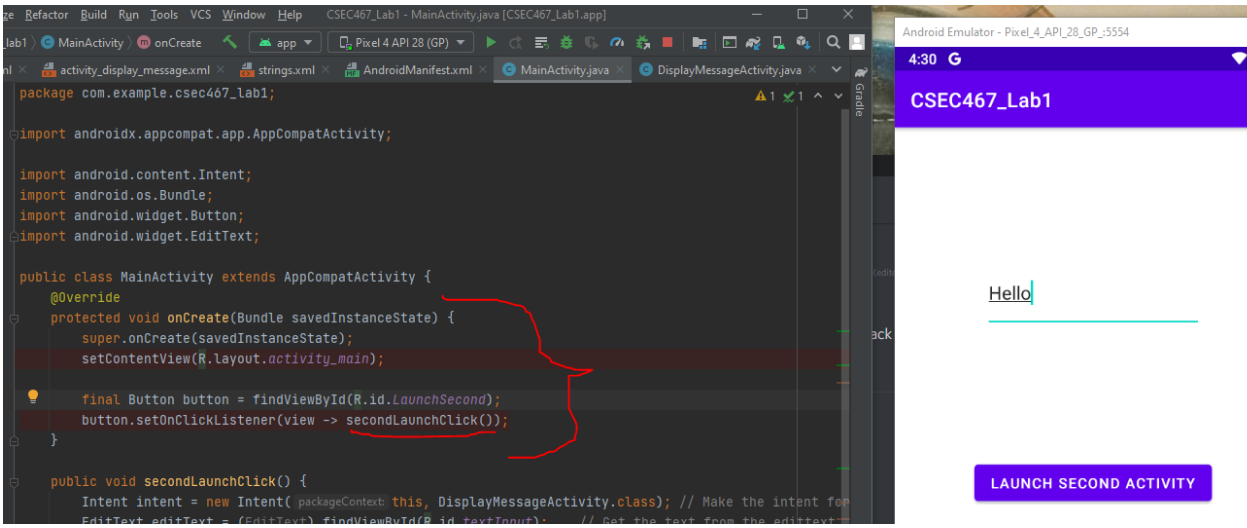


4. Provide the file path at which you were able to find the APK on step 4. Describe the purposes for which Android uses each folder in the file path.
   /data/app/com.example.csec467_lab1-buQUxKvAH3ZELOWyar85Cw==
   /data: Used by android for storing all the information needed by the system.
   /data/app: Used to organize the non-system apps
   /data/app/com.example.csec467_lab1-buQUxKvAH3ZELOWyar85Cw==: Contains the APK for the app package.

5. Describe how completing step 5 differed from completing step 4.
   In step 5, we used the Google play enabled device. This means that the device was not rooted and therefore we could not access root through traditional means. Because of this, we are unable to read the information in the folders under /data. As a result we were unable to locate the APK, or extract it.
6. Provide three reasons why Google may opt to restrict root user access on devices that have Google Play services installed.
   1. By restricting root user access you reduce the chances that an ignorant user might accidentally ruin an important config file and as such brick their device.
   2. With restricted root access, it makes it harder for attackers to try to gain access to the root user by cutting off the ability to use social engineering to gain the user's root password.
   3. With denied access to root privileges, any app that might be malicious would have to declare what permissions it needs giving you the chance to deny it. If there was root access this could be circumvented.

Screenshots:

1. Data Flow
   a. Draw the interface and establish button OnClickListener:



b. Retrieve Inputted Text

```
public void secondLaunchClick() {
    Intent intent = new Intent( packageContext: this, DisplayMessageActivity.class); // Make the intent for
    EditText editText = (EditText) findViewById(R.id.textInput);    // Get the text from the edittext
    String message = editText.getText().toString(); // Convert the edit text text field to a string
    intent.putExtra( name: "MESSAGE", message);    // Attach the string from textInput to the new activi
    startActivity(intent);  // Create the actual activity based off the intent   intent: "Intent { cmp=
}
```

Variables

this = {MainActivity@10964}

Hello

LAUNCH SECOND ACTIVITY

c. Draw new interface

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_display_message);

    Intent intent = getIntent();    // Get the intent that sparked the creation of this activity
    String message = intent.getStringExtra( name: "MESSAGE");

    final Button view_text = findViewById(R.id.view_input);
    view_text.setOnClickListener(view -> displayToast(message));
}

public void displayToast(String message){
    makeText( context: this, message, Toast.LENGTH_SHORT).show();
}
```

CLICK TO VIEW INPUT

d. Retrieve passed information from intent

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_display_message);

    Intent intent = getIntent();    // Get the intent that sparked the creation of this activity
    String message = intent.getStringExtra( name: "MESSAGE");

    final Button view_text = findViewById(R.id.view_input);
    view_text.setOnClickListener(view -> displayToast(message));
}

public void displayToast(String message){
    makeText( context: this, message, Toast.LENGTH_SHORT).show();
}
```

CLICK TO VIEW INPUT

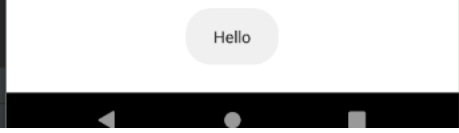
e. Attach onClickListener to button view

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_display_message);

    Intent intent = getIntent();    // Get the intent that sparked the creation of this activity
    String message = intent.getStringExtra( name: "MESSAGE");

    final Button view_text = findViewById(R.id.view_input);
    view_text.setOnClickListener(view -> displayToast(message));
}

public void displayToast(String message){
    makeText( context: this, message, Toast.LENGTH_SHORT).show();
}
```

CLICK TO VIEW INPUT

f. Display text using toast

```
    public void displayToast(String message){
        makeText( context: this, message, Toast.LENGTH_SHORT).show();
    }
}
```



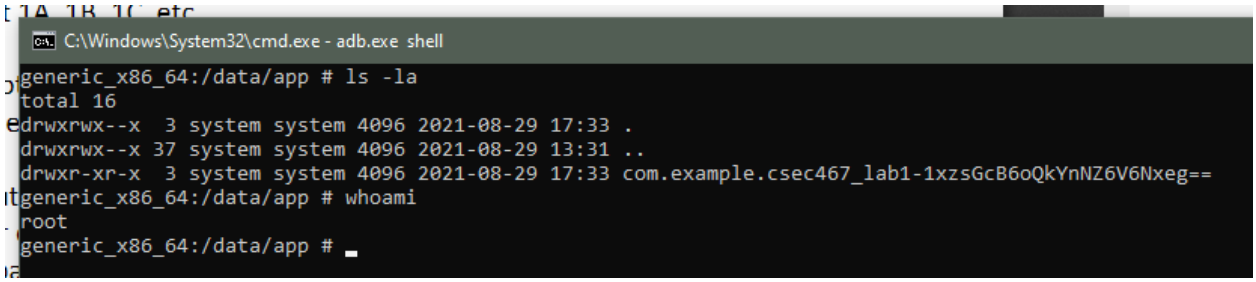
2. Transfer Data Between activities
   a. Send

```
public void secondLaunchClick() {
    Intent intent = new Intent( packageContext: this, DisplayMessageActivity.class);
    EditText editText = (EditText) findViewById(R.id.textInput);    // Get the t
    String message = editText.getText().toString(); // Convert the edit text tex
    intent.putExtra(EXTRA_MESSAGE, message);    // Attach the string from textIn
    startActivity(intent);  // Create the actual activity based off the intent
}
```

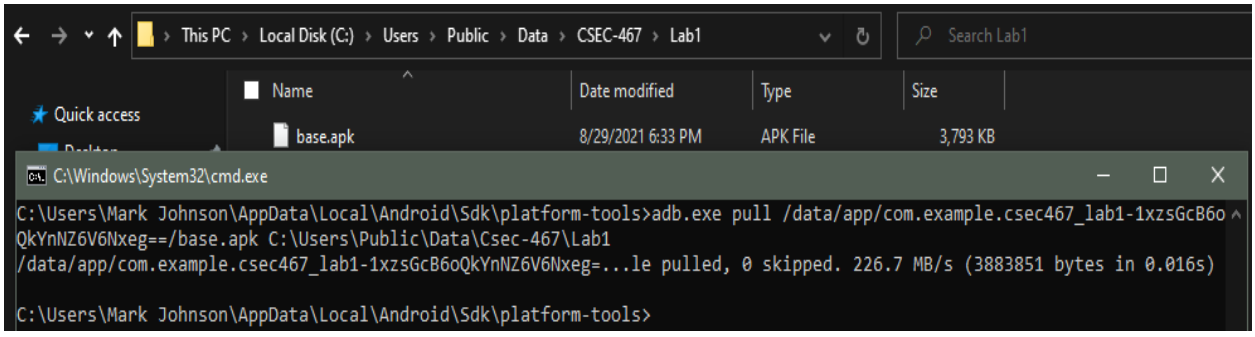
   b. Receive

```
Intent intent = getIntent();    // Get the intent that sparked the creation of this activity
String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
```

3. Ls of folder containing the .apk containing folder. Accessed via root. Did *not* use GenyMotion.



4. Successful extraction of apk.



5. Failed listing of folder containing the folder that holds the apk