Intent Protection

Background

In Android, we can send data between apps quite easily. For example, when we want to share images on Facebook from a local app, we send the image to the Facebook app over the bundle. Then, the Facebook app reads the image data and posts it. Another example might be that when we want to call someone, we send the phone number to call over the bundle to the calling API.

Sending data between apps is especially important for companies with multiple apps that need to interact in some way.

The problem is that sending data between apps is insecure in Android. For example, if we send data from app A to app B, any app C might have the same application ID of app B. App C could then receive the data that should be received by app B. We should not send important data between applications without taking defensive measures.

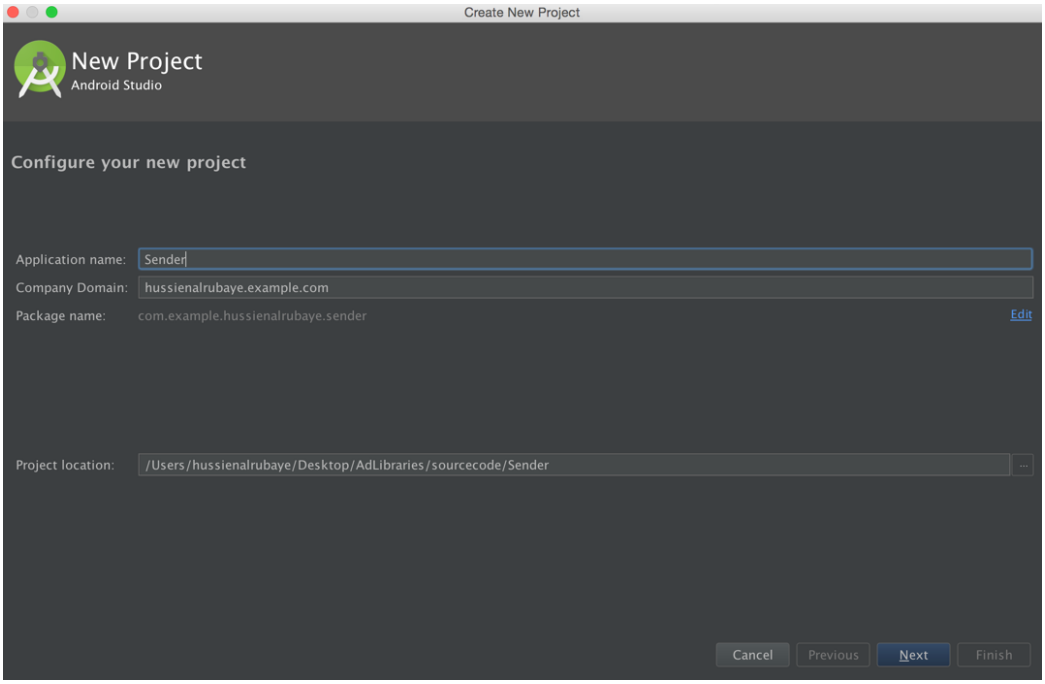
Where and what format to store files in depends on your situation. Depending on how sensitive the data may be, you may even want to take precautions of encrypting it with a key not easily accessible to any possible attackers.

Activity Instructions

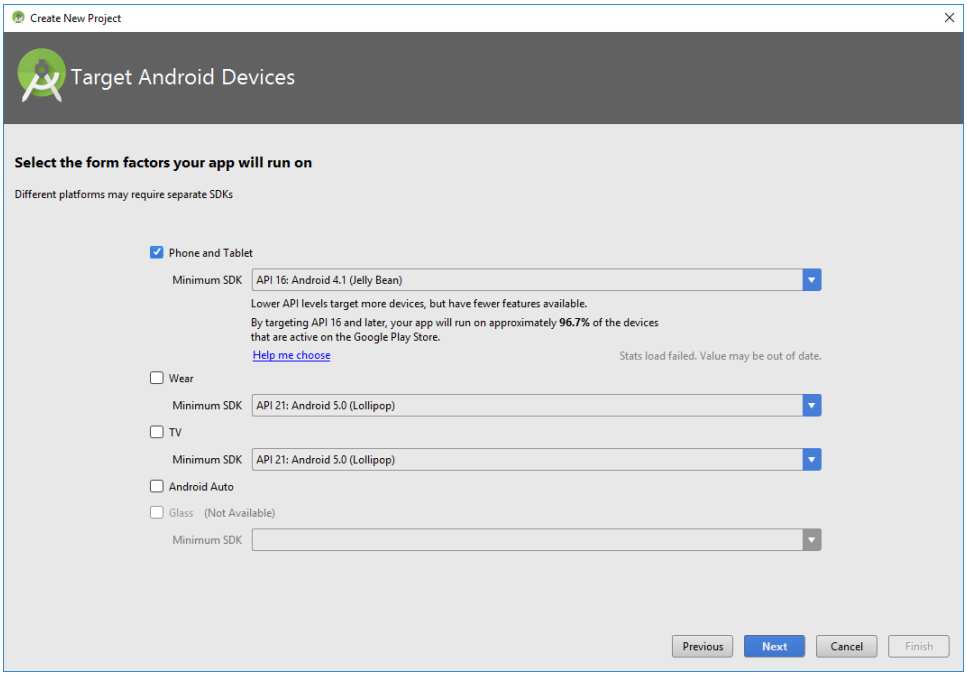
To illustrate the problem, we will:

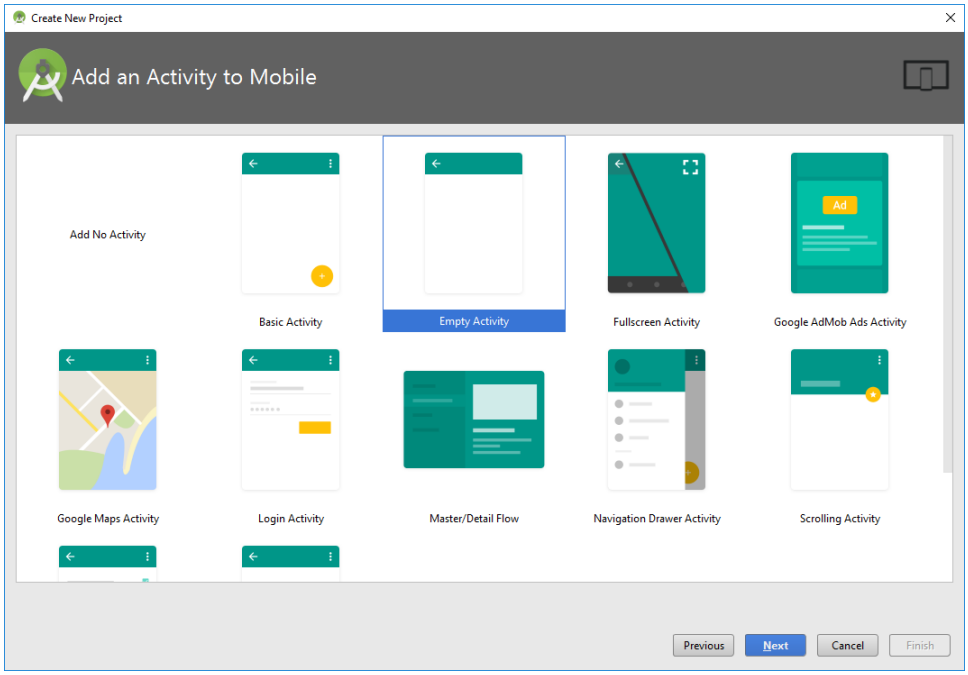
i. Create an app A that sends comment data to another app B
ii. Create the app B to receive the information app A is sending
iii. Create an app C to intercept the comment data meant for app B
iv. Explain techniques to avoid this.

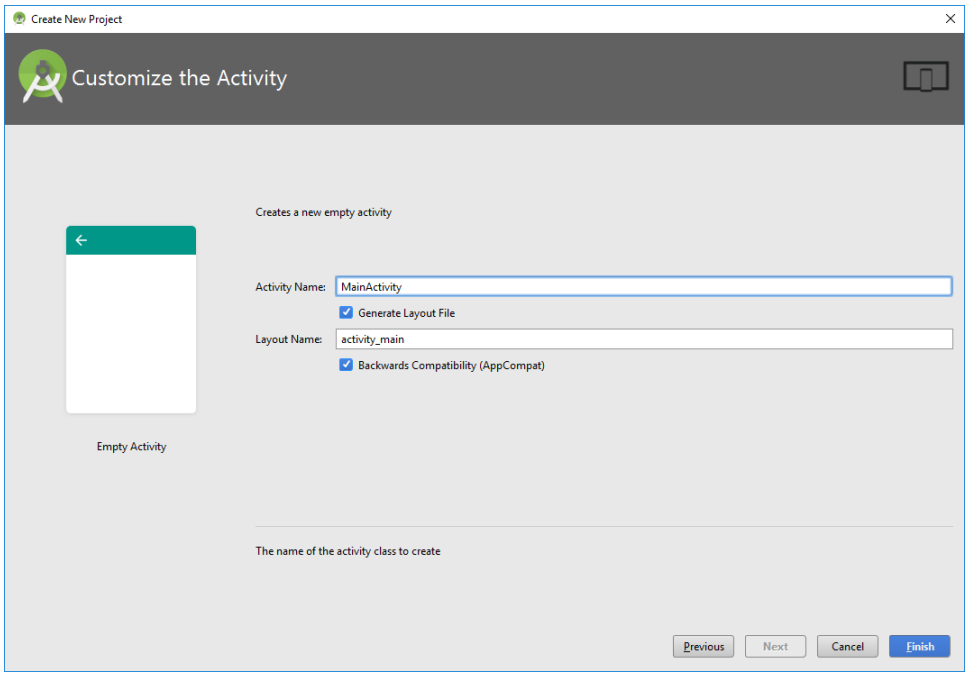
1. Project Creation: Sender App
    a. Follow the screens below to create a new project:



Name the project “Sender”. Remember to save the package name. We will need that later.





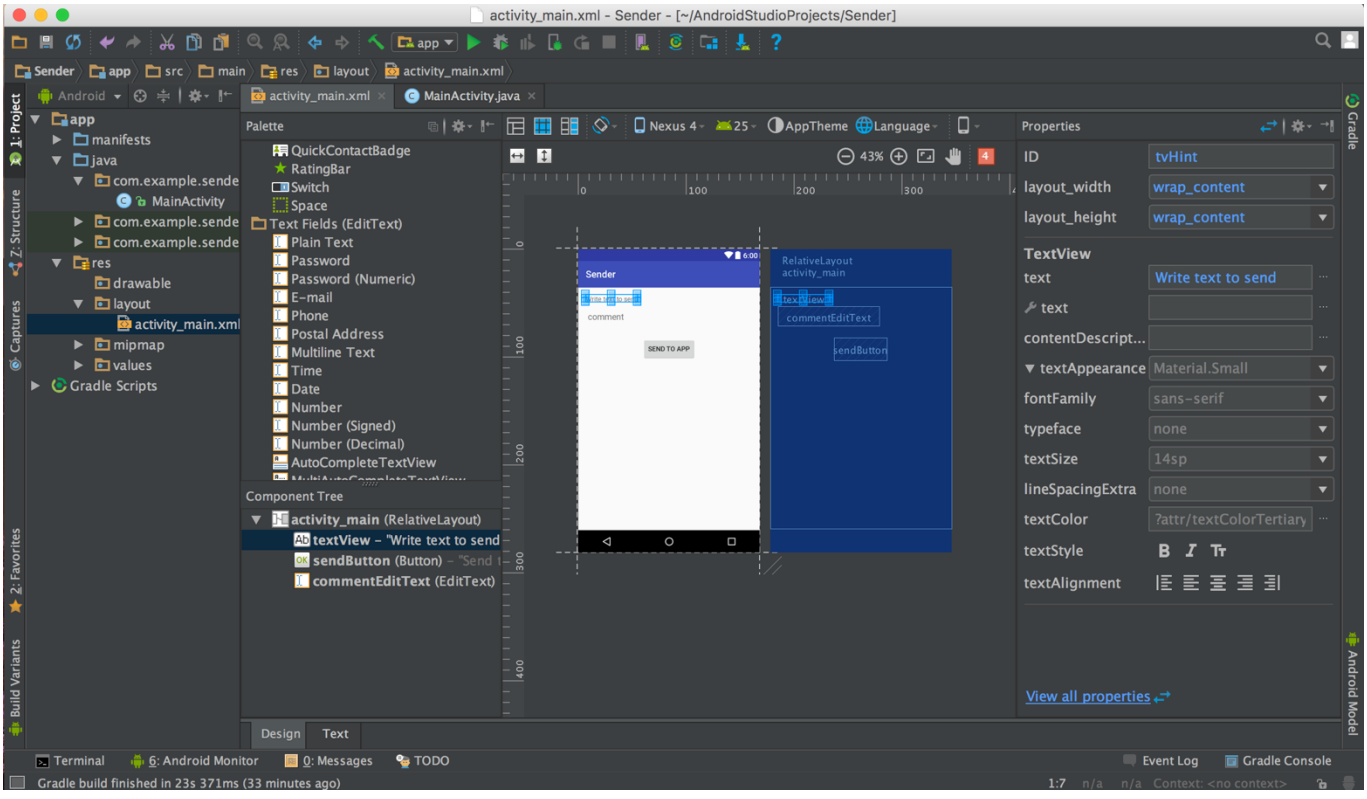


2. Construct User Interface: Sender App
    a. From the Palette tool window, add the following UI controls into the screen layout. Access the screen layout by going to "app/res/layout/activity_main.xml".
        i. Within the existing Relative Layout add:
            - Add the following UI controls:
                1. **Button**:

- Update the following properties in the panel found to the right:
    - `text="Send to app"`
        - It may take a bit of tinkering before Android Studio stops completing the word "app" to "@string/app_name"
    - `id="storeButton"`

2. **TextView**
    - Update the following properties:
        - `id="tvHint"`
3. **EditText**
    - Update the following properties:
        - `id="commentEditText"`
        - `text=""`
        - `hint="comment"`

Following is the hierarchical layout of the controls on the screen:



3. Code: Sender App

Open MainActivity.java, found under "app/java/your_package_name", and add the following code:

a. Add the following code inside the **onCreate** method:

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Define send button
        Button sendButton = (Button)findViewById(R.id.sendButton);

        // init commentEditText
        final EditText commentEditText =
(EditText)findViewById(R.id.commentEditText);

        // button listen to click event
        sendButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // set the package that we want to run
                Intent intent =
getPackageManager().getLaunchIntentForPackage("com.example.receiver");

                    // put the data that we want to send over intent
                    intent.putExtra("Comment",
   commentEditText.getText().toString());

                    // start another app
                    startActivity(intent);
                }
            });
}
```

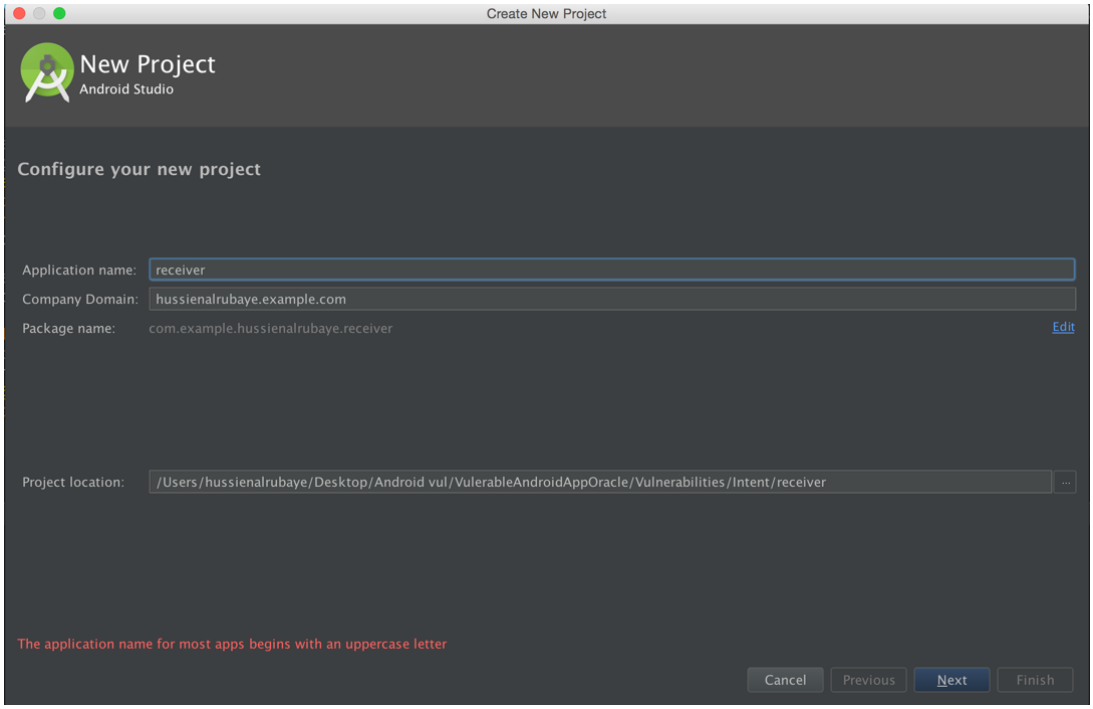
b. Add the following imports to the file, below the package declaration.

```
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
```

The above code achieves the following:

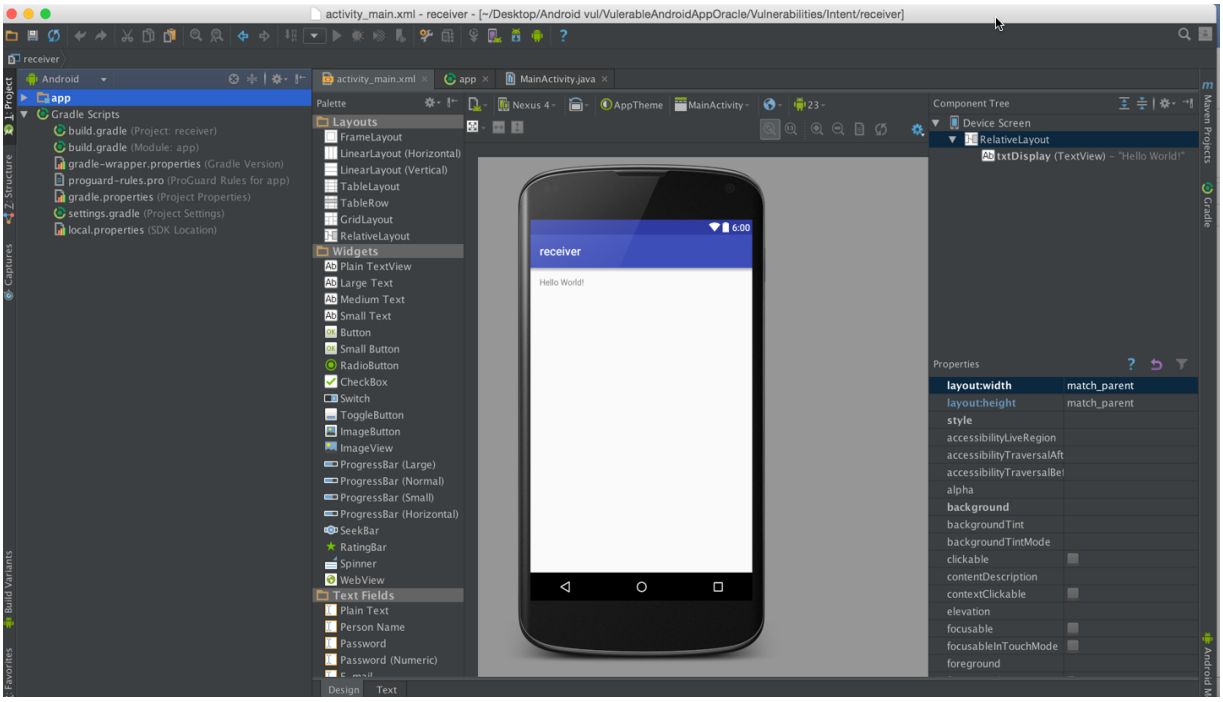
1. Upon creation, the onCreate method executes.
    a. Passes the Bundle named `savedInstanceState` to the superclass `AppCompatActivity`
    b. Sets the ContentView to be the activity_main.xml we worked on earlier.
    c. Initializes the Button and EditText variables
    d. Creates a listener for the send button that upon click, will:
        i. Create an Intent for the package we want to run
        ii. Add a key-value pair to represent the comment data
        iii. Starts the activity using the Intent

1. Project Creation: Receiver App
    a. Create a new project in Android Studio. Name the project "Receiver", and take note of the package name. We will need it later.



2. Construct User Interface: Receiver App
    a. From the Palette tool window, add the following UI controls into the screen layout. Access the screen layout by going to "app/res/layout/activity_main.xml".
        i. Within the existing Relative Layout add:
            - Add the following UI controls from the Palette bar into the design view of the screen layout:
                1. **TextView**:
                    - Update the following properties in the panel found to the right:
                        - `id="txtDisplay"`

The layout might look like this:



3. Code: Receiver App

   Open MainActivity.java, found under "app/java/your_package_name", and add the following code:

   a. Add the following code inside the **onCreate** method:

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

    // Define the display Text view
    TextView txtview = (TextView)findViewById(R.id.txtDisplay);

    // get app the data sent on bundle
    Bundle b = getIntent().getExtras();

    // display the key that have the data
   txtview.setText(b.getString("Comment"));
}
```

   b. Ensure that the list of libraries imported under the package declaration looks something like this:

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;
```

The above code achieves the following:

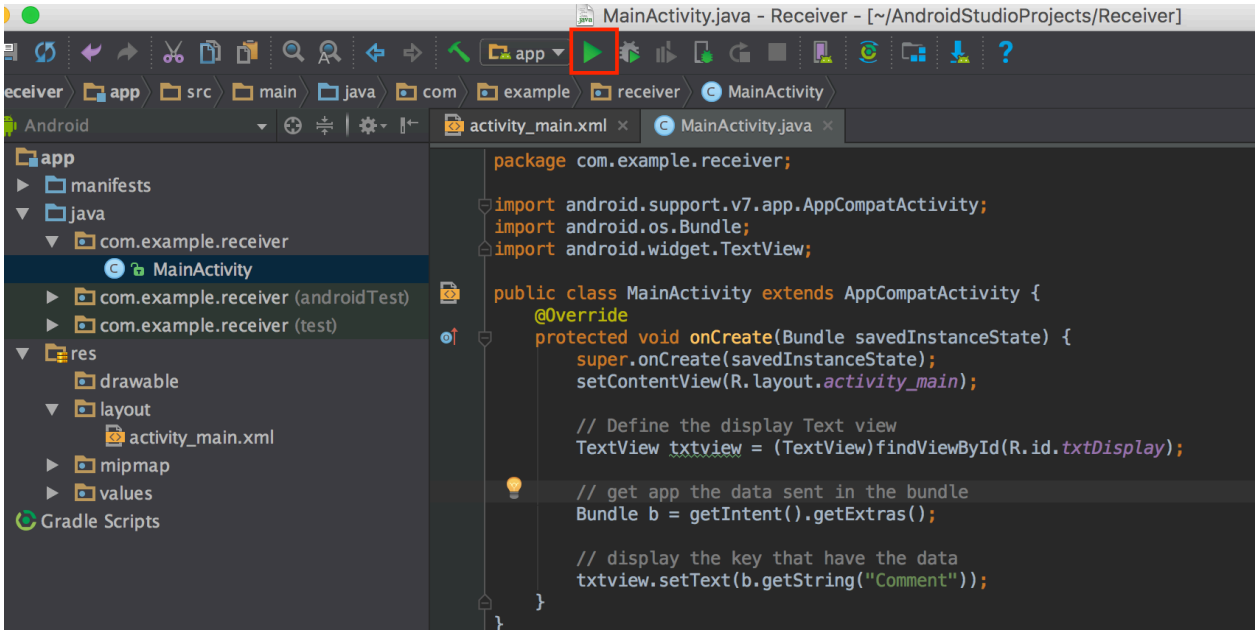
1. Upon creation, the onCreate method executes.
    a. Passes the Bundle named `savedInstanceState` to the superclass `AppCompatActivity`
    b. Sets the ContentView to be the activity_main.xml we worked on earlier.
    c. Initializes the TextView variable
    d. Grabs the data that came bundled
        i. Grabs the value that we can find in the Intent under the key "Comment"
        ii. Sets the view of the TextView to display the value

Run and execute the Receiver app by clicking the button circled in red below.

Exploitation Instructions

We shall see for ourselves how we can intercept the comment data meant for the Receiver

1. Run the Receiver app by clicking the button circled in red below.

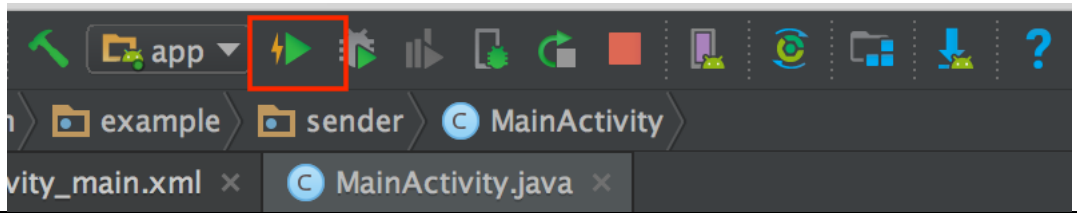


It will stop running and throw an error, but that is okay. We just need to run it that first time to load the app onto the device.

The Sender app will call the Receiver app, and as such, the Receiver app needs to be loaded onto the device before we run Sender. Note that it is important to load the apps onto the same virtual or physical device. We need all three apps to be running on the same device.

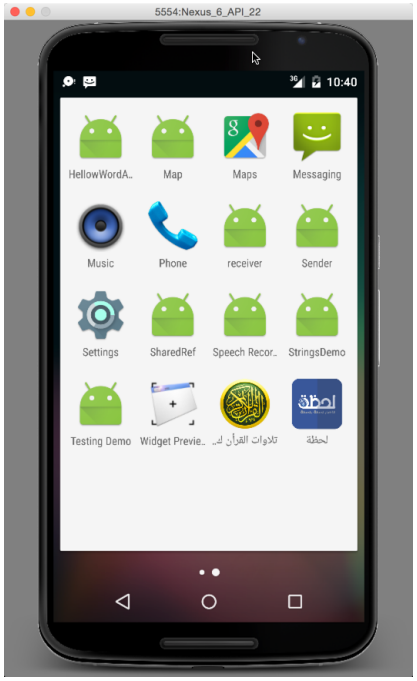
2. Build and run the Sender app. You can leave the emulator running in between these steps.

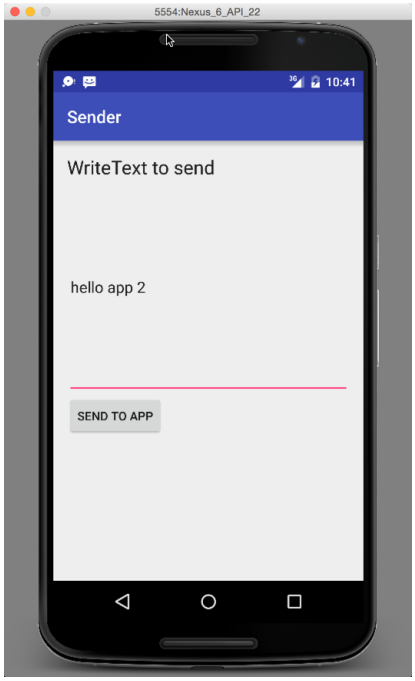
If the button to compile, load, and run has changed to the one depicted below, that simply means that Android Studio will be pushing any updates to the app directly, without rebuilding it from scratch. It is fine to continue.



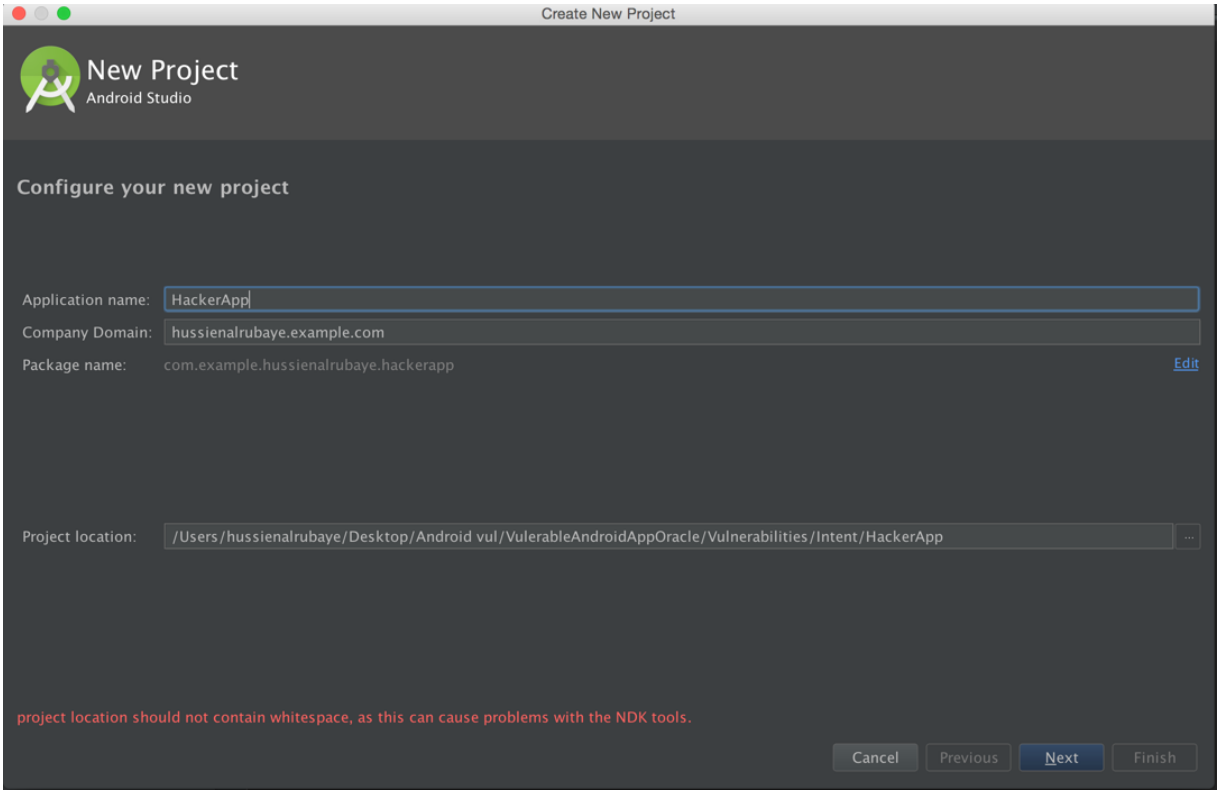
3. Enter text into sender, click the Send to App button. When the Receiver app comes up, it will be displaying the comment you entered in Send to App.

   See below for some images of what this could look like.





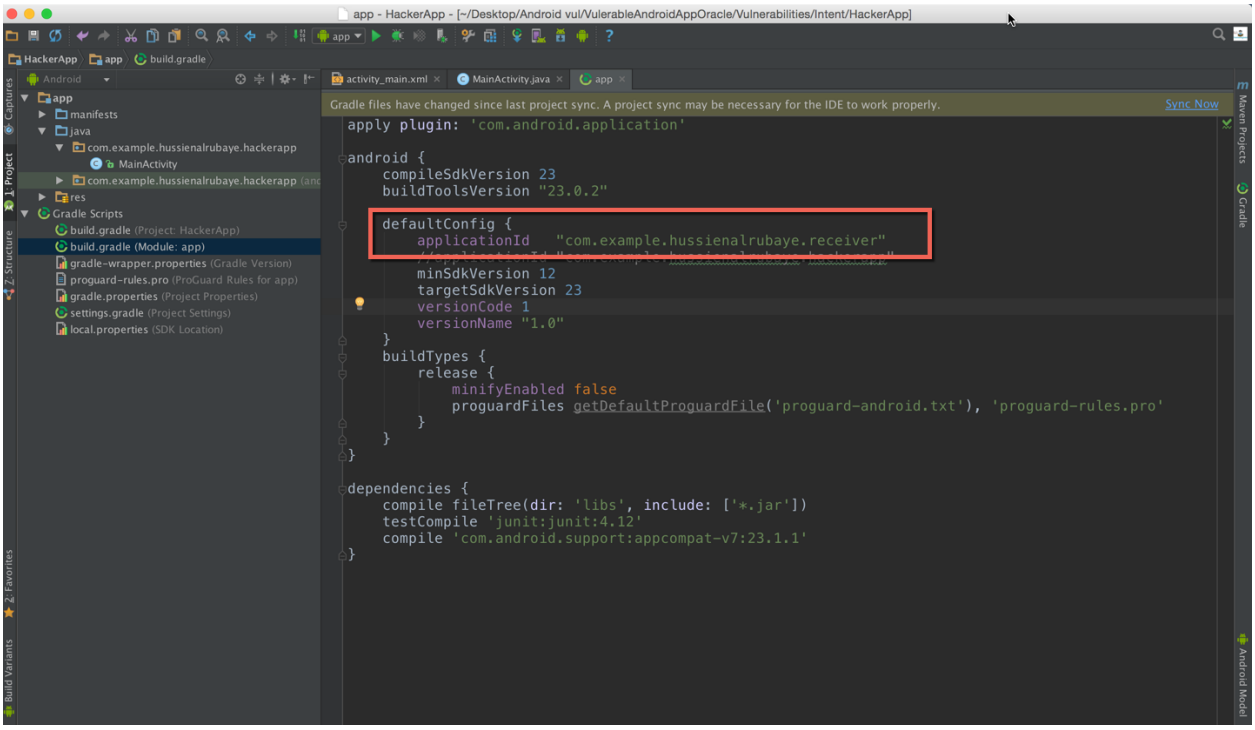
4. <u>Project Creation: Hacker App</u>
   Name the project “HackerApp”. Save the package name for later.



5. Construct User Interface: Hacker App
    b. From the Palette tool window, add the following UI controls into the screen layout. Access the screen layout by going to "app/res/layout/activity_main.xml".
        ii. Within the existing Relative Layout add:
            - Add the following UI controls from the Palette bar into the design view of the screen layout:
                1. **TextView**:
                    - Update the following properties in the panel found to the right:
                        - `id="txtDisplay"`

    If the app template is Empty, it usually already has a "Hello, World" TextView present. We can use that.

6. Go to the Gradle build script for the specific module, found under "app/Gradle Scripts/", in the build.gradle (Module: app) file. Change the applicationId to the applicationId the Receiver app has. Since they now have the same applicationId, it could receive the data that should be received by as the Receiver.



7. Code: Hacker App

Open MainActivity.java, found under "app/java/your_package_name", and add the following code:

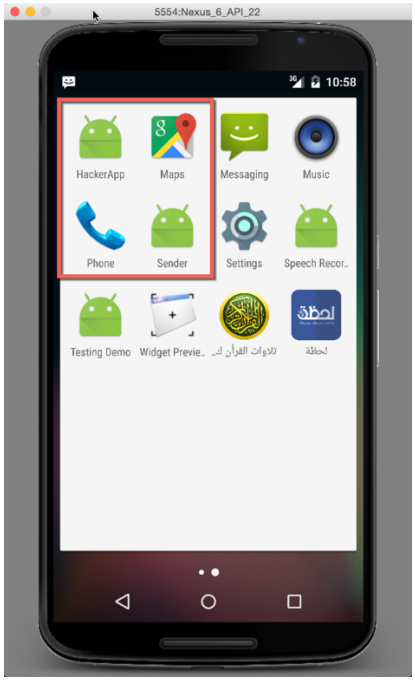
a. Add the following code inside the **onCreate** method:

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

    // init the textView to display data
    TextView txtDisplay = (TextView)findViewById(R.id.txtDisplay);
    String dataBundle = "";

    // get the data sent in bundle from the app
    Bundle bundle = getIntent().getExtras();

    // loop through all keys in the bundle
    for (String key : bundle.keySet()) {
        // get object by key - we define it as object because we are not sure
of type
        Object value = bundle.get(key);

        // get all keys
        dataBundle += String.format("%s %s (%s)", key, value.toString(),
value.getClass().getName());
     }
    txtDisplay.setText(dataBundle);
}
```

b. The list of imports below the package declaration should look like this:

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;
```

8. Build and run the apps to intercept the data.
   i. Run and build the HackerApp as described above. Again, the app will crash. This is because it is reliant upon receiving a data bundle from the Intent. In this situation, however, we are running the app ourselves. The bundle ends up being empty (more correctly known as Null), and will crash when we try to iterate over it.
   ii. After we've loaded HackerApp onto our device, we go back to the home screen and run Sender. Write any comment and click to send.
   iii. The "Click to Send" button brings up HackerApp instead of Receiver. See that it has indeed intercepted the data meant for Receiver.



Defense

To ensure that apps will not be able to read the data, we encrypt the data before even sending it in a bundle. After receiving the data, we decrypt it. This means that even if apps intercepted the data, they would not be able to read it.