Activities Access

## Background

Android activity navigation is quite like how Internet navigation works. In web-based applications, many pages are public. For example, we could access many pages by adding its name after the domain name, like so: <http://host/home.html>. In Android, we could do the same thing -- we could navigate to any activity using “am” package commands.

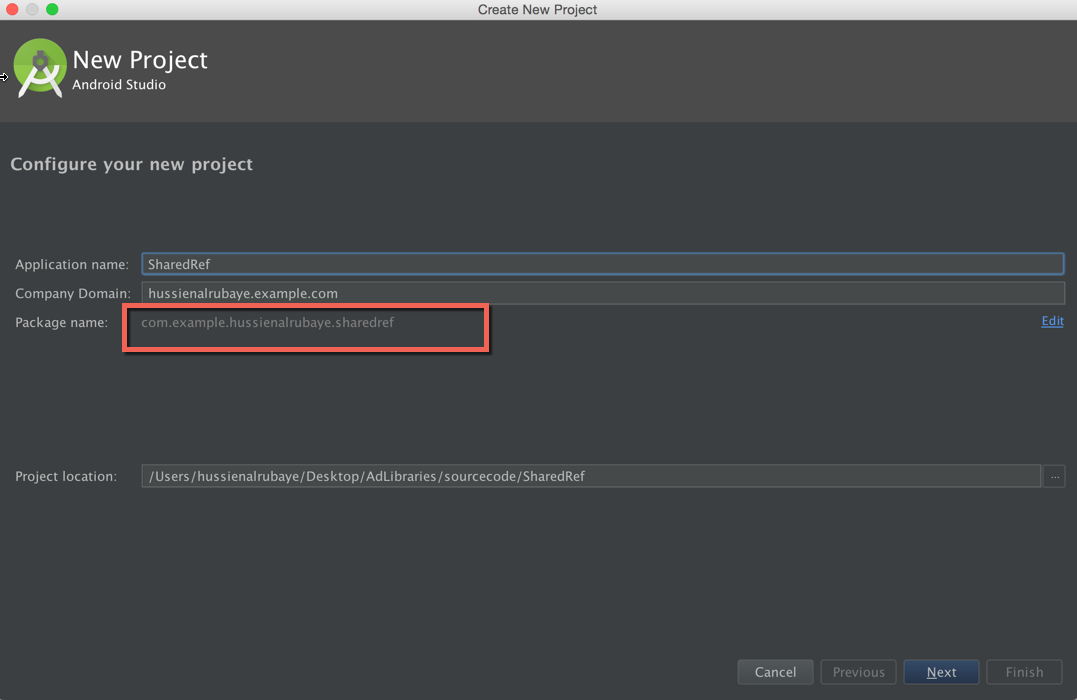
This could be quite the security problem. There are certain activities whose access should be restricted. Take a bank app that has two activities: the first activity for login and the second activity for making transactions the user must login to do transactions.

The security weakness here is that hackers could possibly circumvent login and navigate to the restricted-access activities by using “am” package commands.

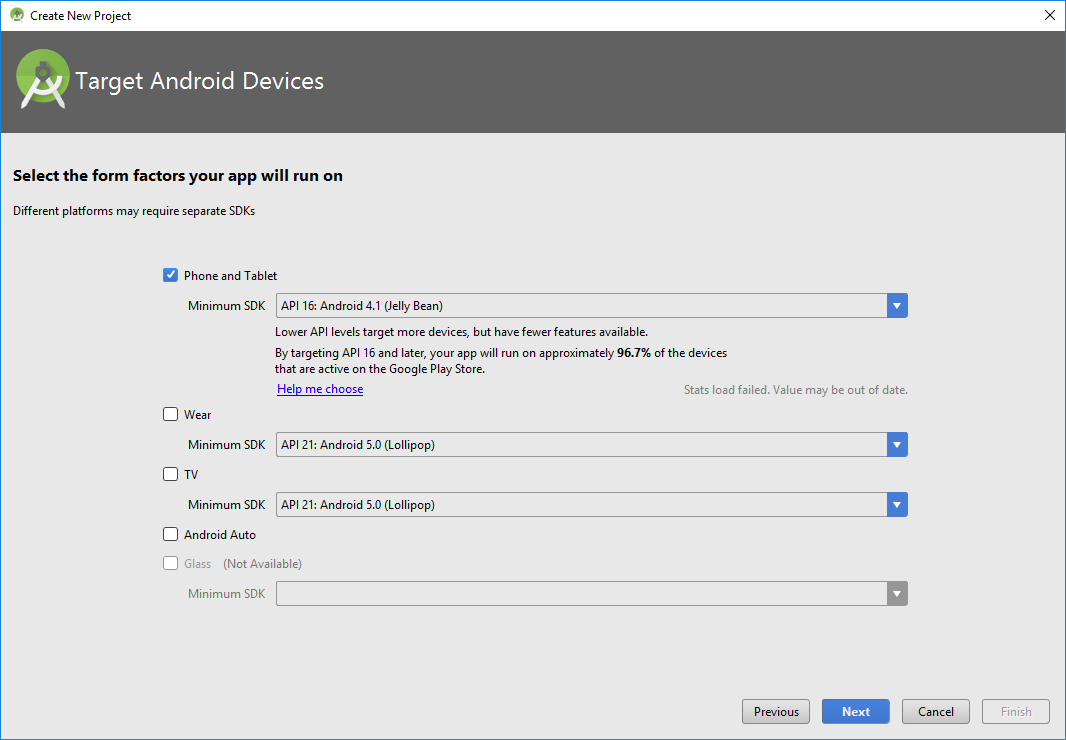
## Activity Instructions

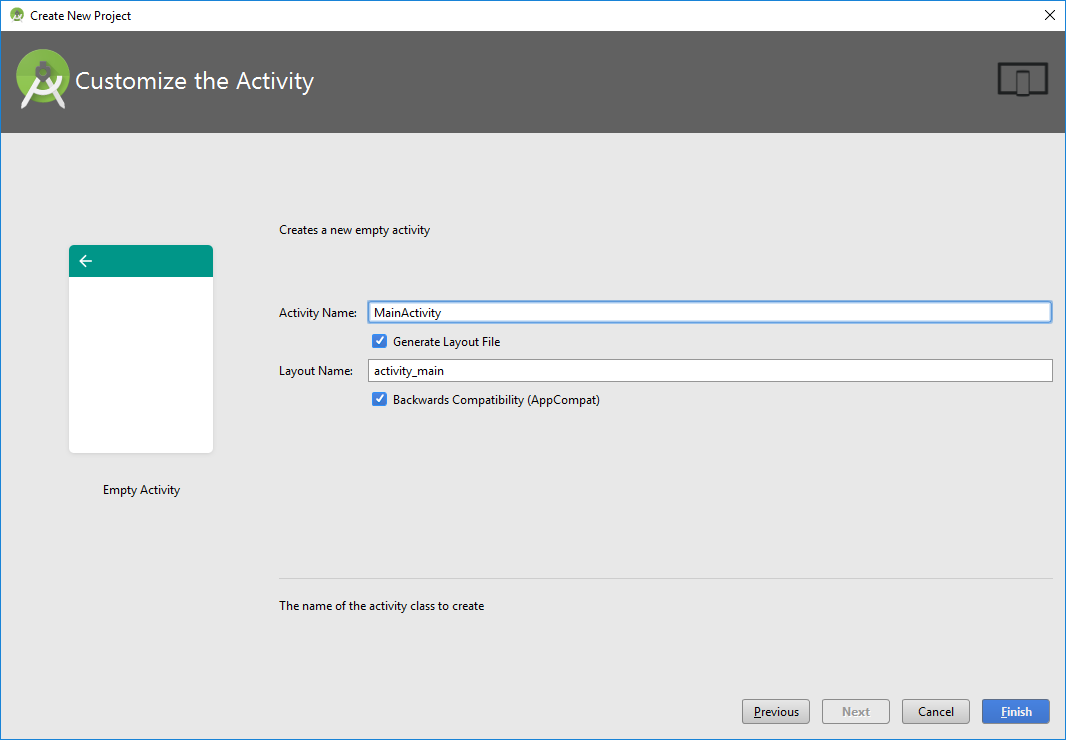
We will illustrate the problem by creating an app and exploiting it ourselves.

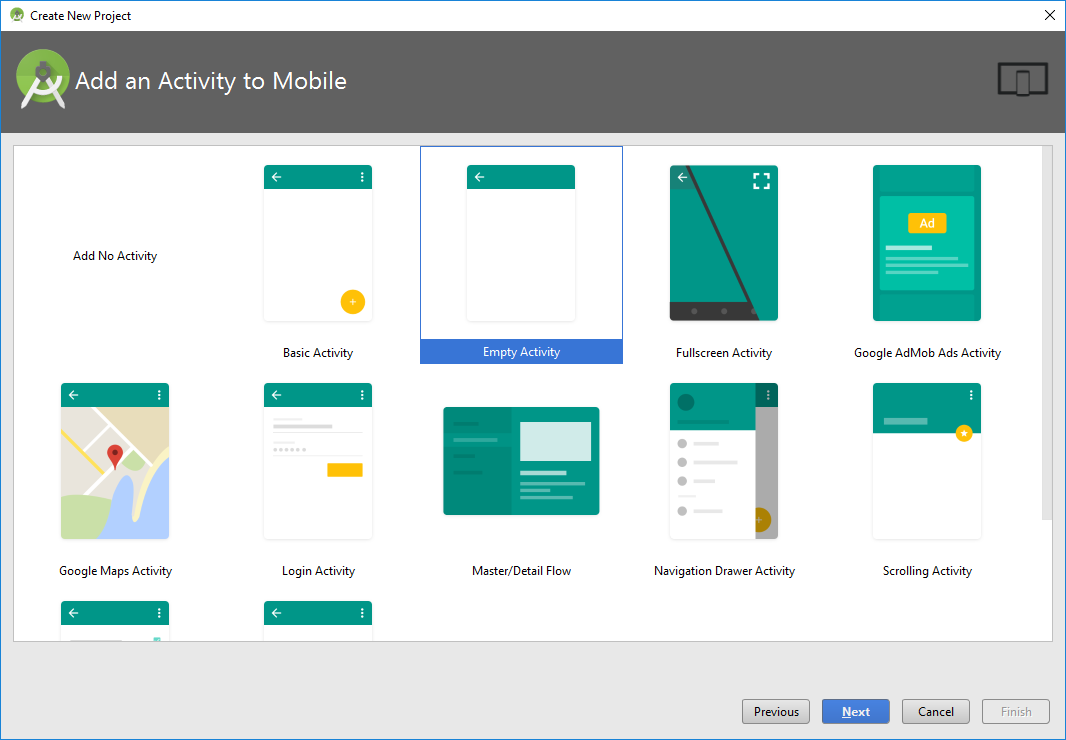
1. Create an app that asks the user to enter a username and password. Upon correct entry of login credentials, the user will be redirected to the second page to change their password.
2. Show how an attacker might access the activity to change the user password without login.
3. Explain techniques we might use to defend ourselves.
4. Project Creation
   1. Follow the screens below to create a new project:



Name the project “SecureActivityAccess”.







1. Add an Activity
   1. Add a new activity called “Main2Activity.java” by opening “ActivityMain.java”, found under “app/java/package\_name\_here/MainActivity”. Under the panel at the top, click File > New > Activity > Basic Activity.
2. Construct User Interface
   1. 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”.

* + 1. Within the existing Relative Layout:
* Add the following UI controls:
  + - * 1. **TextView**:
    - Update the following properties:

id="textView”

text="Username”

* + - * 1. **TextView**
* Update the following properties:

id="password"

text=”Username”

* + - * 1. **EditText**

Update the following properties:

id=”usernameEditText"

text=””

inputType=”textPersonName”

* + - * 1. **EditText**

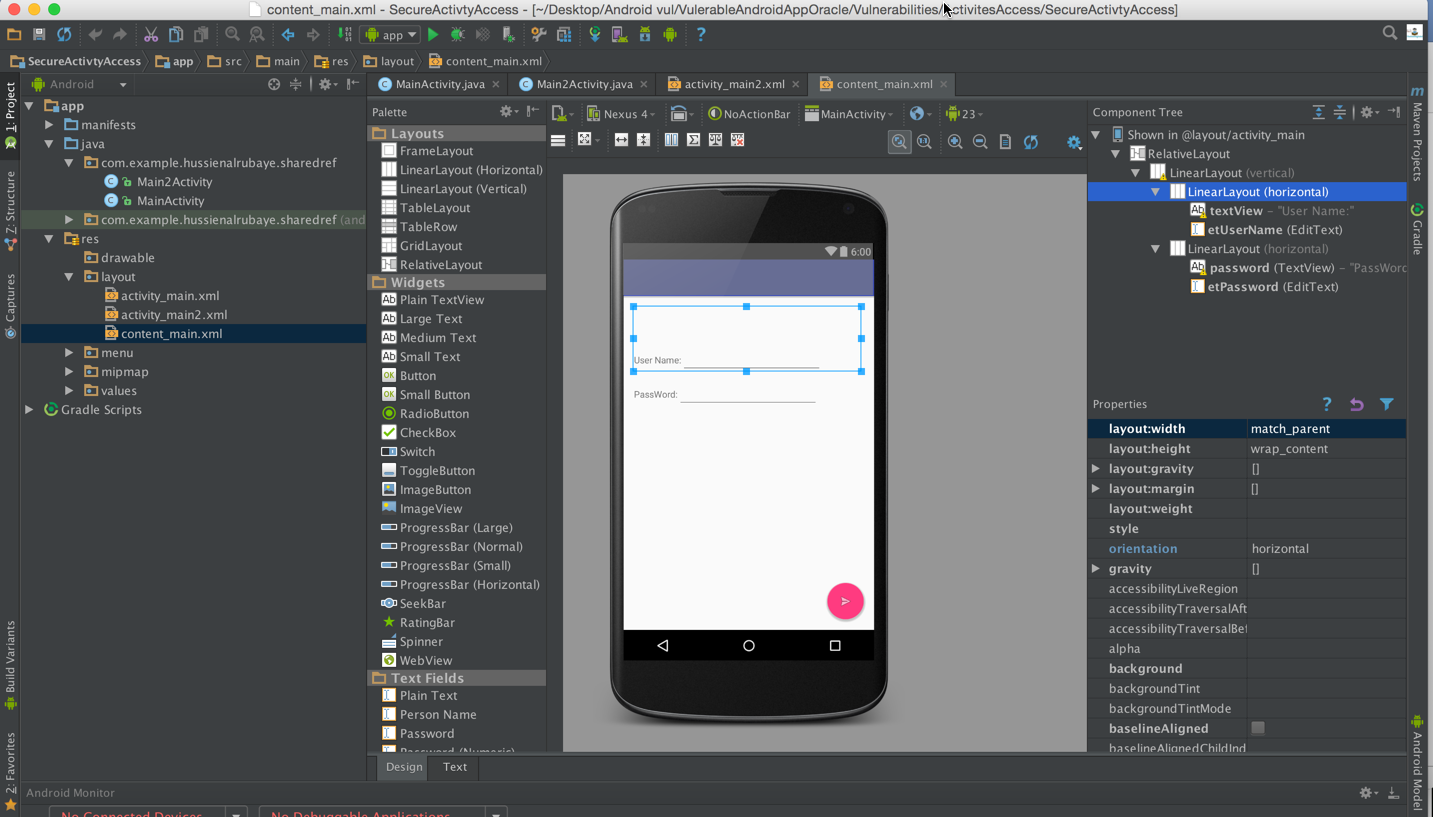
Update the following properties:

id=”passwordEditText"

hint=”Password”

inputType=”textPassword”

This is what the layout should look like:



* 1. Construct the layout for the second activity we created the same way, only with the “Username” and “Password” labels changed to “Change Password” and “Repeat Password”. Remember to change both EditTexts to take input of type password.

1. Code

Open MainActivity.java, found under “app/java/your\_package\_name”, and add the following code:

* 1. Declare the following variables:

|  |
| --- |
| **public class** MainActivity **extends** AppCompatActivity {  *// key for user name* **public final** String **UserName** = **"admin"**;  *// key for password* **public final** String **Password** = **"admin"**;  *// shared references instance to access to virtual file* SharedPreferences **sharedpreferences**;  *// input text name* EditText **usernameEditText**;  *// input text password* EditText **passwordEditText**; |

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

|  |
| --- |
| @Override **protected void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.***activity\_main***);  Toolbar toolbar = (Toolbar) findViewById(R.id.***toolbar***);  setSupportActionBar(toolbar);  *// initialize user name instance with the real input in xml* **etUserName** = (EditText)findViewById(R.id.usernameEditText);  *// initialize password instance with the real input in xml* **etPassword** =(EditText)findViewById(R.id.passwordEditText);  *// initialize shared references* FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.***fab***);  *// listen to floating button when click* fab.setOnClickListener(**new** View.OnClickListener() {  @Override  **public void** onClick(View view) {   **if**(**UserName**.equals(**etUserName**.getText().toString()) && **Password**.equals(**etPassword**.getText().toString()))  *//display message saved* {  Toast.*makeText*(MainActivity.**this**, **"Data is Saved"**, Toast.***LENGTH\_LONG***).show();  Intent intent = **new** Intent(getApplicationContext(), Main2Activity.**class**);   startActivity(intent);  }  }  });} |

* 1. Add the following imports to the file, below the package declaration.

|  |
| --- |
| **import** android.content.Intent; **import** android.content.SharedPreferences; **import** android.os.Bundle; **import** android.support.design.widget.FloatingActionButton; **import** android.support.v7.app.AppCompatActivity; **import** android.support.v7.widget.Toolbar; **import** android.view.View; **import** android.widget.Toast; **import** android.widget.EditText; |

Open Main2Activity.java, and add the following code:

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

|  |
| --- |
| @Override **protected void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.***activity\_main2***);  Bundle b=getIntent().getExtras();  } |

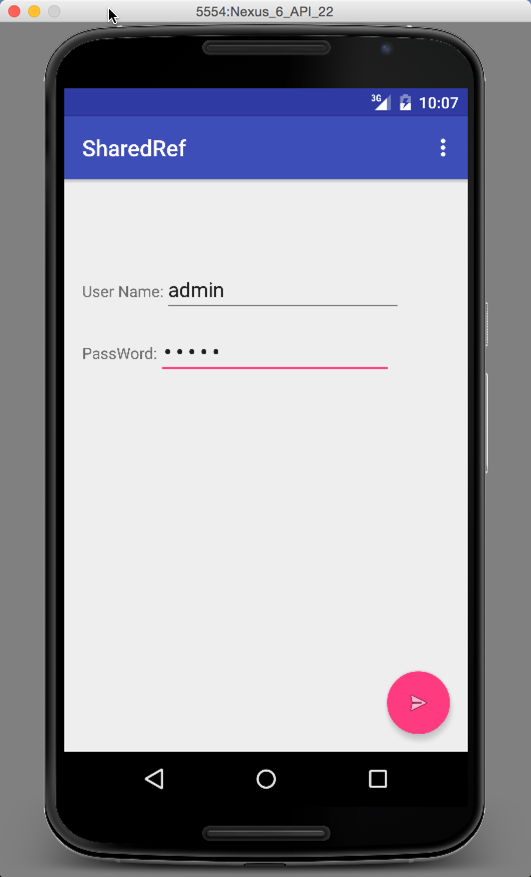
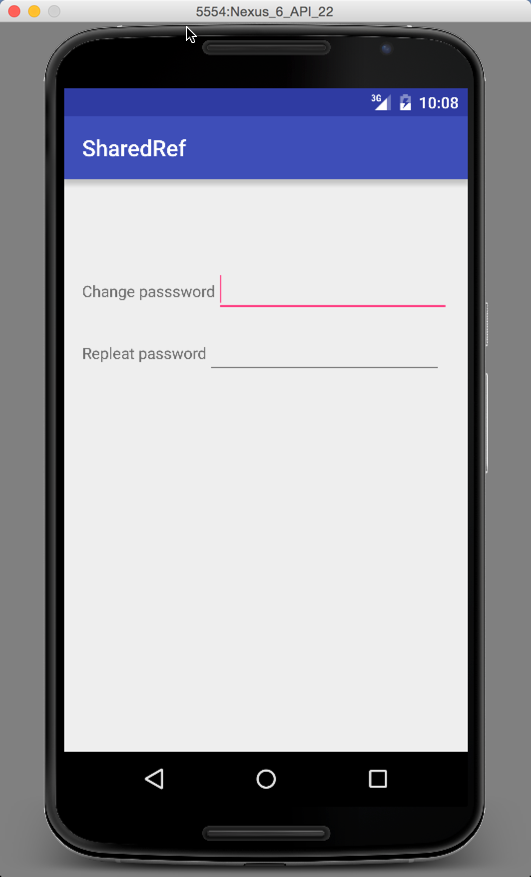
The above code achieves the following:

1. Upon creation of the MainActivity class, the **onCreate** method executes.
   1. Passes the Bundle named savedInstanceState to the superclass AppCompatActivity
   2. Initializes the variables declared in the first part with their corresponding layout objects.
   3. Creates a listener for the store button that upon click, will:
      1. Compare the credentials entered to the strings we have saved. If the credentials match what we have, we create an Intent with the second activity.
      2. Start the second activity.
2. Upon creation of the Main2Activity class, the **onCreate** method executes.
   1. Passes the Bundle named savedInstanceState to the superclass AppCompatActivity
   2. Sets the content view to be the layout we designed for the second activity.
   3. Grabs the bundle that came with the Intent.

## Exploitation Instructions

We shall see for ourselves how we can view the login credentials.

1. Run the app. Enter the login credentials of “admin” for username and “admin” for password (demonstration purposes only).



1. Using adb shell, view the saved preferences file by:
   1. Open Terminal or Command Prompt.
   2. Run the following commands.
      1. On Mac OS X:

cd Library/Android/sdk/platform-tools

On Windows:

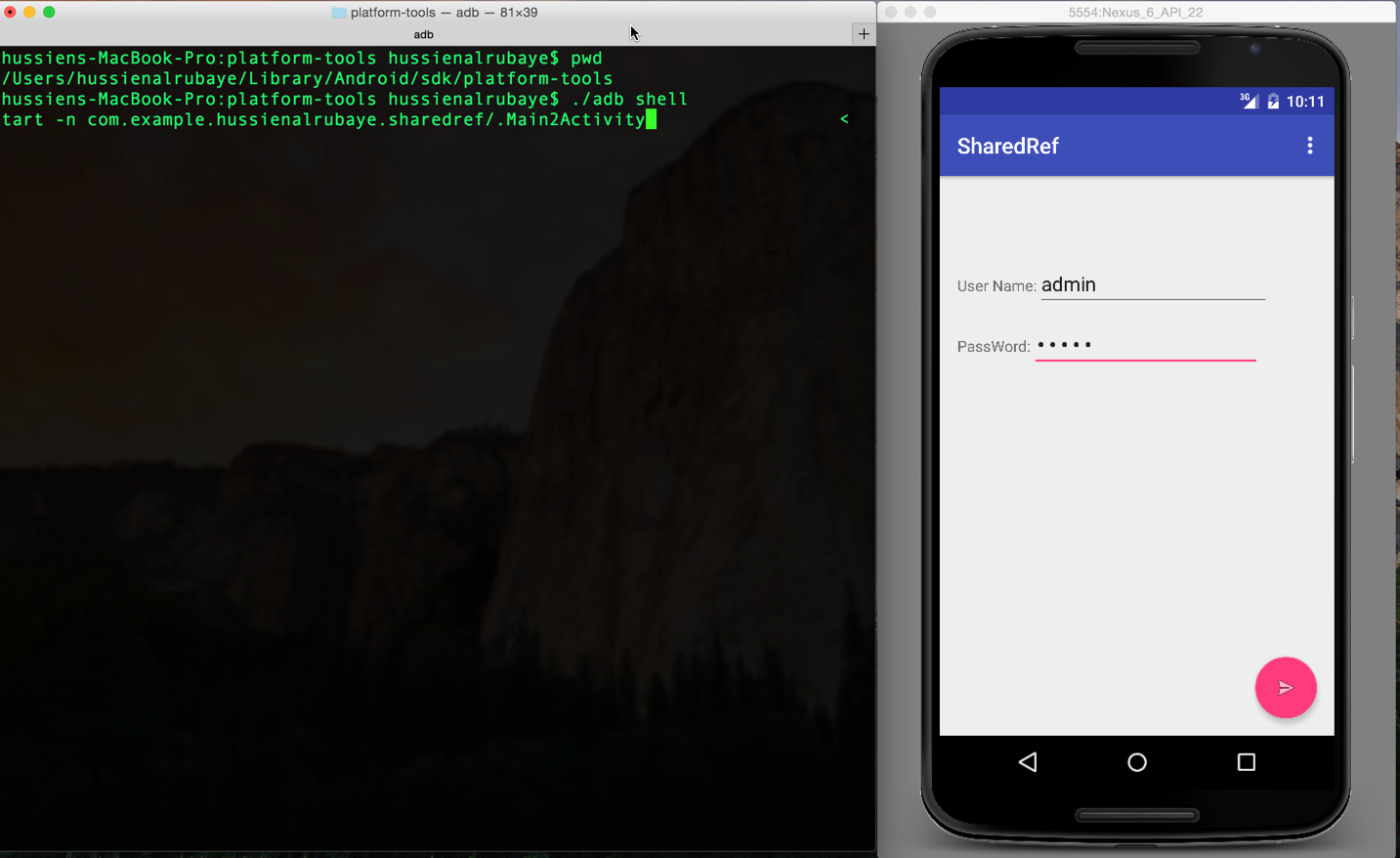
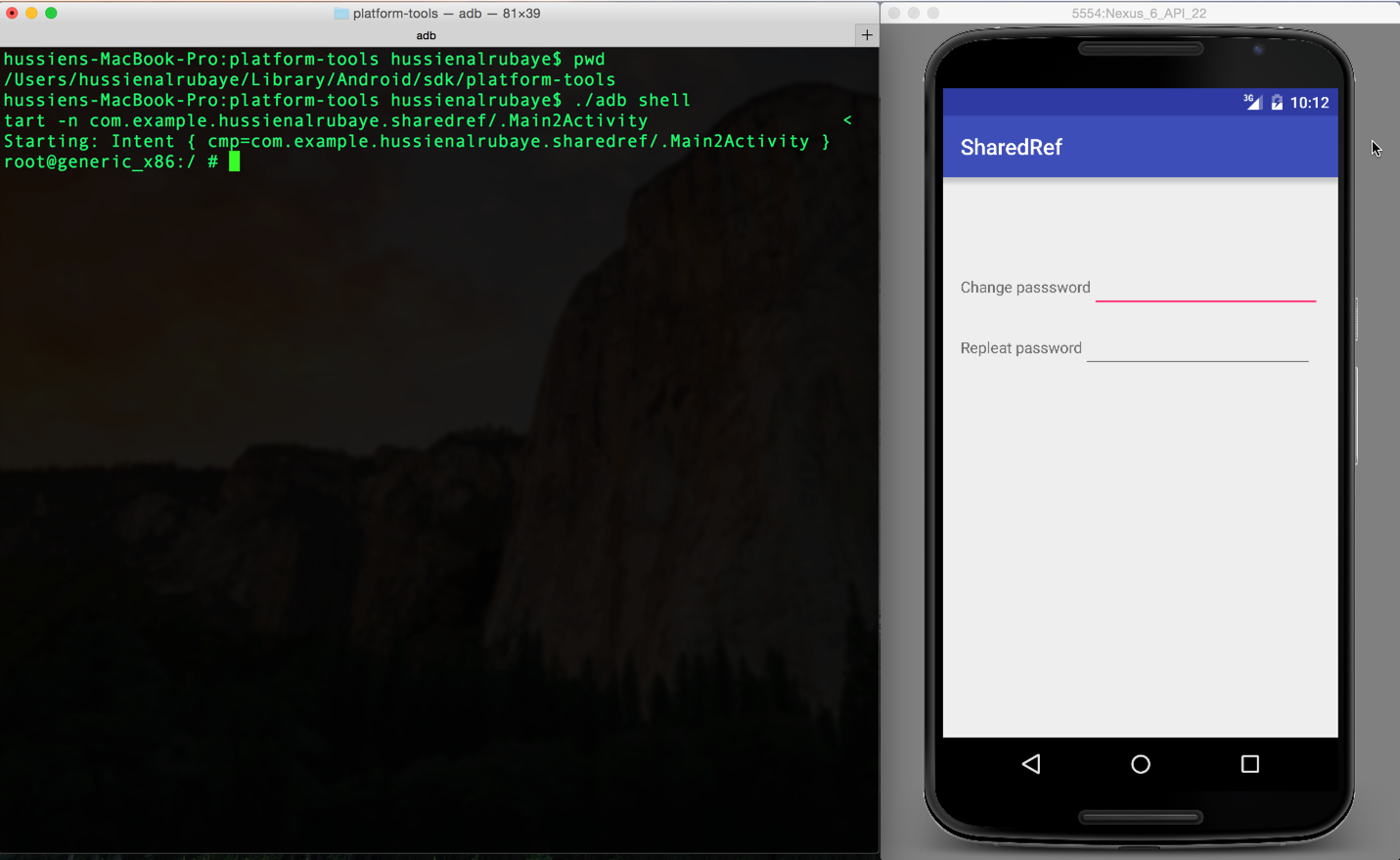
cd C:\Users\YOUR\_USERNAME\_HERE\AppData\Local\Android\sdk\platform-tools

* + 1. From here, it doesn’t matter what platform you are running this on. We simply needed to find the Android/sdk/platform-tools directory.

./adb shell

* + 1. am start –n package\_name/.ActivityName

1. Once you have executed the commands above, you will be sent to the activity that is supposed to be after login only. We will be able to change the password without login.



## Defense

To fix this problem, we will send the key associated with the value over the intent to change password activity. In the second activity, we will then read the key and make sure the value is correct. If it is correct, we can start the password-changing activity. Otherwise, we will dismiss the activity. Then, when we run the “am” command without the key to open Main2Activity, it will not open.

1. Code

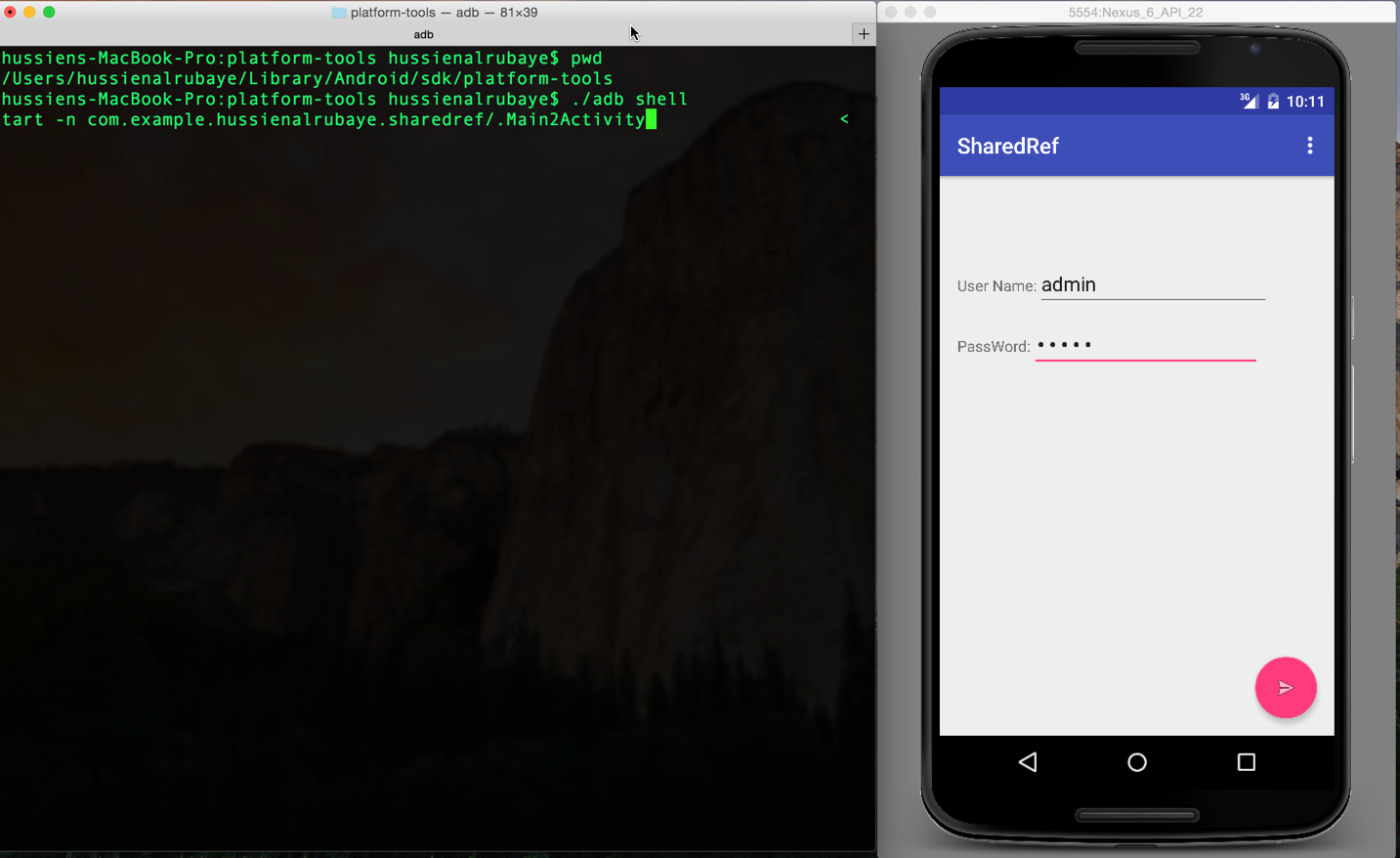
The **onCreate** method of the MainActivity class is very like the one we see in the previous part. The only thing that has changed is the highlighted line: we include a key-value pair with the intent we are passing to the startActivity function.

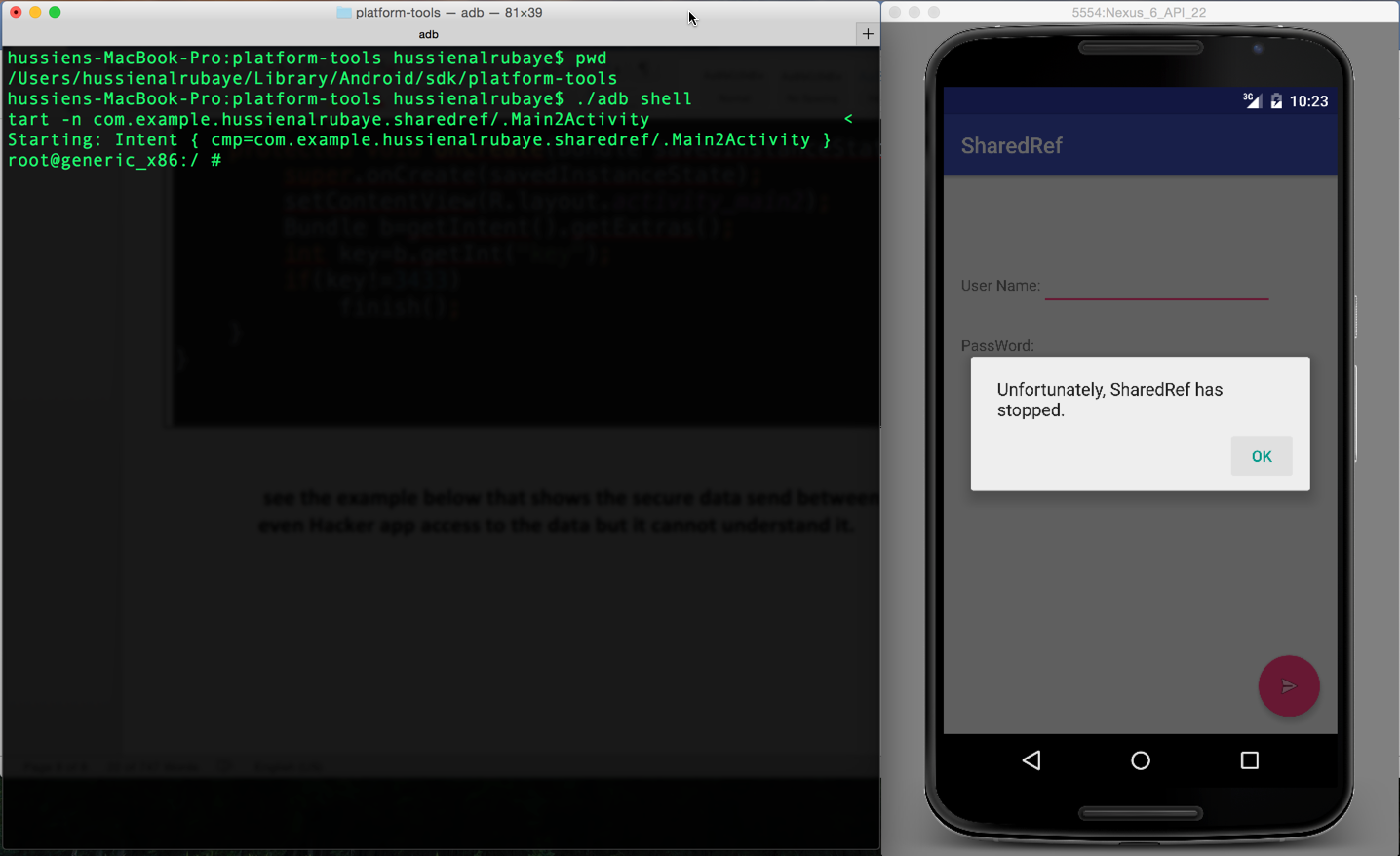
|  |
| --- |
| @Override **protected void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.***activity\_main***);  Toolbar toolbar = (Toolbar) findViewById(R.id.***toolbar***);  setSupportActionBar(toolbar);  *// initialize user name instance with the real input in xml* **etUserName**=(EditText)findViewById(R.id.etUserName);  *// initialize password instance with the real input in xml* **etPassword**=(EditText)findViewById(R.id.etPassword);  *// // initialize shared references* FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.***fab***);  *// listen to floating button when click* fab.setOnClickListener(**new** View.OnClickListener() {  @Override  **public void** onClick(View view) {   **if**(**UserName**.equals(**etUserName**.getText().toString()) && **Password**.equals(**etPassword**.getText().toString()))  *//display message saved* {  Toast.*makeText*(MainActivity.**this**, **"Data is Saved"**, Toast.***LENGTH\_LONG***).show();  Intent intent = **new** Intent(getApplicationContext(), Main2Activity.**class**);  intent.putExtra(**"key"**, 3433);  startActivity(intent);  }   }  }); } |

Now, we change the **onCreate** method of the Main2Activity class to check the intent it is passed for the key-value pair. If it is incorrect, or non-existent, then the Main2Activity class will simply not start.

|  |
| --- |
| **public class** Main2Activity **extends** AppCompatActivity {  @Override  **protected void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.***activity\_main2***);  Bundle b = getIntent().getExtras();  **int** key = b.getInt(**"key"**);  **if** (key != 3433)  finish();  } } |

1. When we run the same commands in adb again, we will get the following screens:



****

As we can see, an attacker would no longer be able to access the “change password” screen without the key-value pair.