Data Storage

Background

Android provides several options for developers to store data locally using different formats such as files, SharedPreferences and SQLite. Data stored using any of these formats will be stored in the same path where the application is stored on the phone which is:

"`data/data/package_name`"

If we explore this path we can see that there are three folders which are used to store local data in Android, these folders are:

- Files
- Databases
- SharedPreferences

There are two different kinds of storage divisions present in an Android system: internal and external. Internal memory is a part of the device, and will always be available to the apps that store data there. The only apps that will be able to access the files here are the ones who stored them there – this is a simple access restriction enforced by the Android system. Upon uninstall of an app, Android will remove all the related files here.

With external memory, it is assumed that the files stored here will not always be available. External memory might mean that the memory in question is a micro SD card and could be removed at any time. Files here are world-readable, so do not store sensitive data here. Files saved here are removed upon uninstall of app, but only if saved from the `getExternalFilesDir()` function.

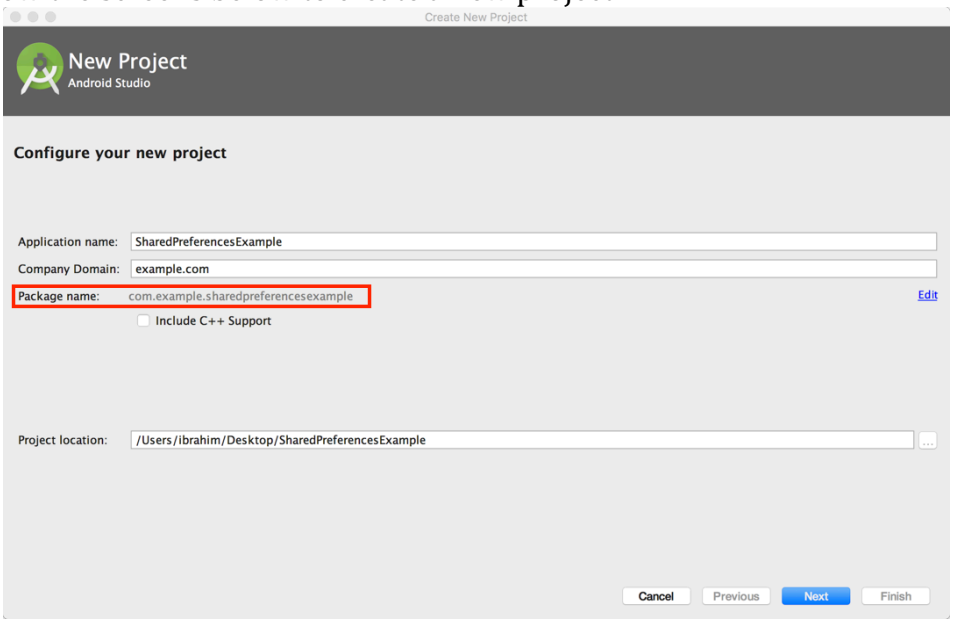
Where and what format to store files in depends on your situation. Depending on how sensitive the data may be, you may even want to take precautions of encrypting it with a key not easily accessible to any possible attackers.

Activity Instructions

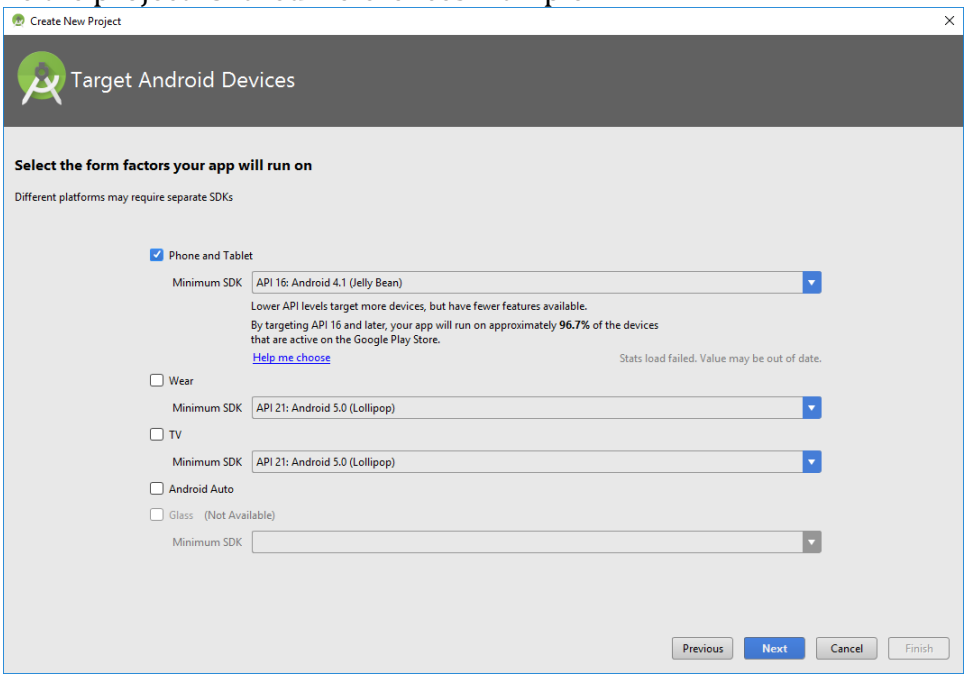
Activity illustrates the problem by doing these things.

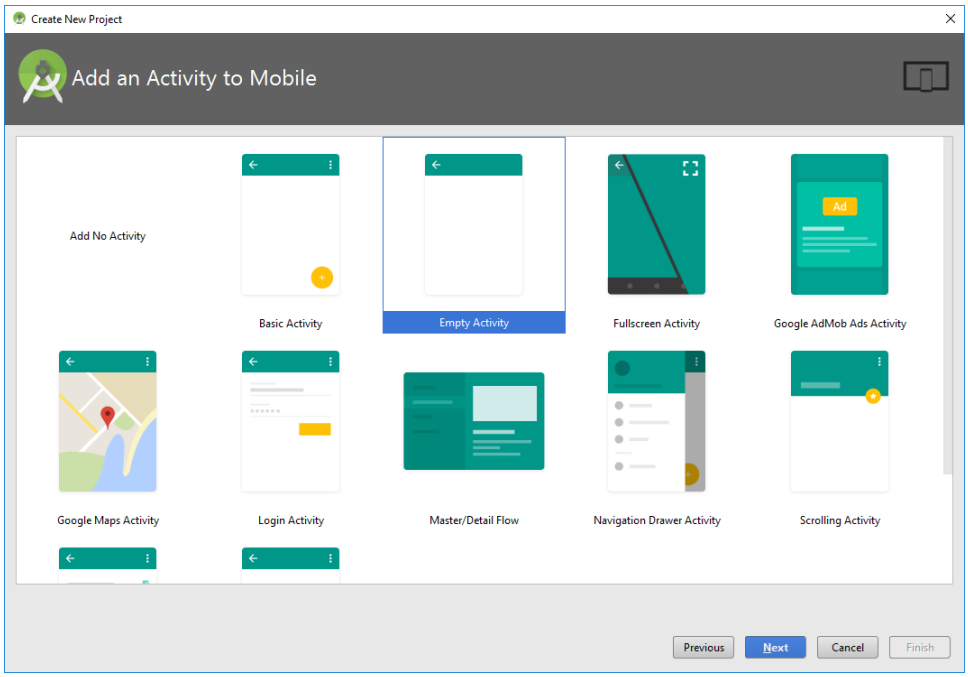
i. Create an app that stores usernames and passwords using SharedPreferences
ii. Read the data without having the proper permissions.
iii. Explain techniques to avoid this.

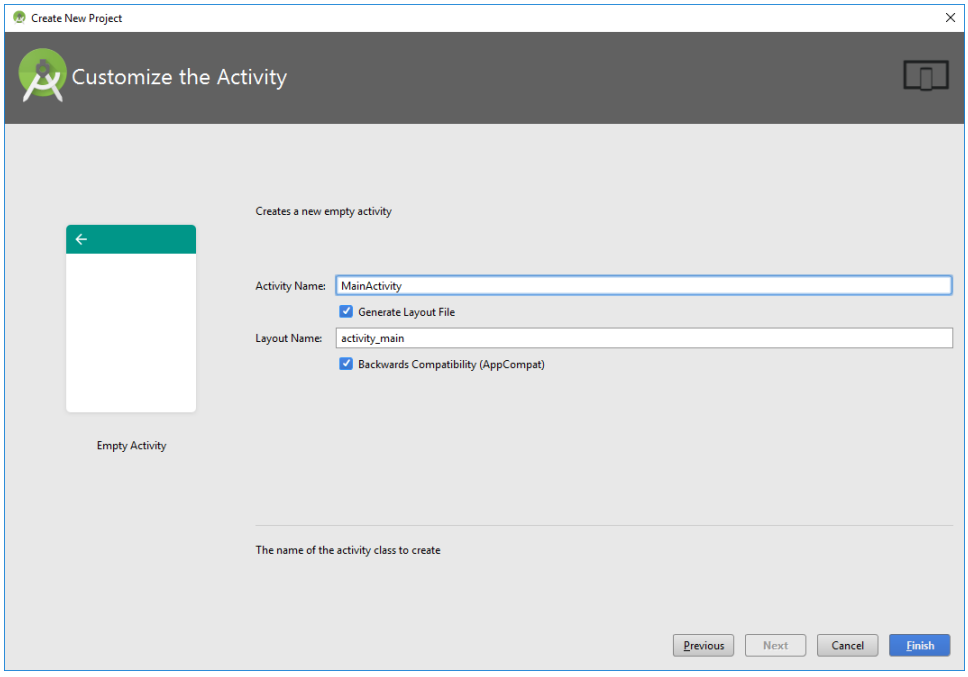
1. Project Creation
   a. Follow the screens below to create a new project:



Name the project “SharedPreferencesExample”.







2. Construct User Interface
   a. From the Palette tool window, add the following UI controls into the screen layout. You could also edit the design portion of the layout purely through the UI Android Studio provides you. See the accompanying video for how to do this.

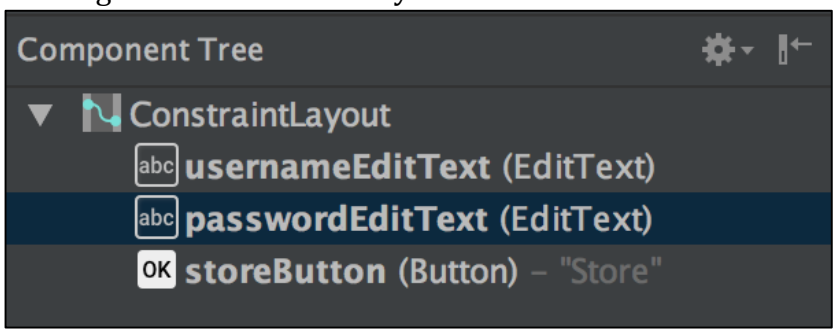
Access the screen layout by going to “app/res/layout/activity_main.xml”.

i. Change the layout type to Constraint Layout, and the properties in the properties panel to the left to the following:
   - `layout_width="match_parent"`
   - `layout_height="match_parent"`

ii. Within the existing Relative Layout add:
   - Add the following UI controls:
     1. **Button**:
        - Update the following properties:
          - `text="Store”`
          - `layout_width="wrap_content"`
          - `layout_height="wrap_content"`
          - `id="storeButton”`
          - `layout_constraintTop_toBottomOf="@+id/passwordEditText"`
          - `layout_marginTop="8dp"`
     2. **TextView**
        - Update the following properties:
          - `id="usernameEditText"`
          - `layout_width="368dp"`
          - `layout_height="wrap_content"`
          - `layout_constraintLeft_toLeftOf="parent"`
          - `layout_constraintRight_toRightOf="parent"`
          - `layout_constraintTop_toTopOf="parent"`
          - `layout_marginEnd=”8dp”`
          - `layout_marginLeft=”8dp”`
          - `layout_marginRight=”8dp”`
          - `layout_marginStart=”8dp”`
          - `layout_marginTop=”8dp”`
          - `ems=”10”`
          - `hint=”Username”`
          - `inputType=”textPersonName”`
     3. **TextView**
        - Update the following properties:
          - `id=”passwordEditText"`
          - `layout_width="368dp"`
          - `layout_height="wrap_content"`
          - `layout_constraintLeft_toLeftOf="parent"`
          - `layout_constraintRight_toRightOf="parent”`
          - `layout_constraintTop_toTopOf=" @+id/usernameEditText”`
          - `layout_marginEnd=”8dp”`
          - `layout_marginLeft=”8dp”`
          - `layout_marginRight=”8dp”`
          - `layout_marginStart=”8dp”`

- `layout_marginTop="8dp"`
- `ems="10"`
- `hint="Password"`
- `inputType="textPassword"`

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



Following is the rendering of controls on the screen:

3. Code

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

a. Declare the following variables:

```
public class MainActivity extends AppCompatActivity {

    public static final String MY_PREFERENCES = "MyPrefLogin";
    public static final String USERNAME = "UsernameKey";
    public static final String USER_PASSWORD = "PasswordKey";
    SharedPreferences mSharedPreferences;
    EditText mUsernameEditText, mPasswordEditText;
    Button mStoreButton;
```

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

```
@Override
protected void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // initialize  username EditText
    mUsernameEditText = (EditText)findViewById(R.id.usernameEditText);

    // initialize  password EditText  instance
    mPasswordEditText = (EditText)findViewById(R.id.passwordEditText);

    // initialize button instance
    mStoreButton = (Button) findViewById(R.id.storeButton);

    // initialize SharedPreferences
    mSharedPreferences = getSharedPreferences(MY_PREFERENCES,
Context.MODE_PRIVATE);

    // Click listener for the button
    mStoreButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            // store data
            // enable start editing file
            SharedPreferences.Editor editor = mSharedPreferences.edit();

            // add user name
            editor.putString(USERNAME, mUsernameEditText.getText().toString());
            // add password
            editor.putString(USER_PASSWORD,
mPasswordEditText.getText().toString());

            // store the update data
            editor.commit();
            //display message stored
            Toast.makeText(MainActivity.this, "Your data has been stored
successfully!", Toast.LENGTH_LONG).show();
        }
    });
}
```

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

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

The above code achieves the following:

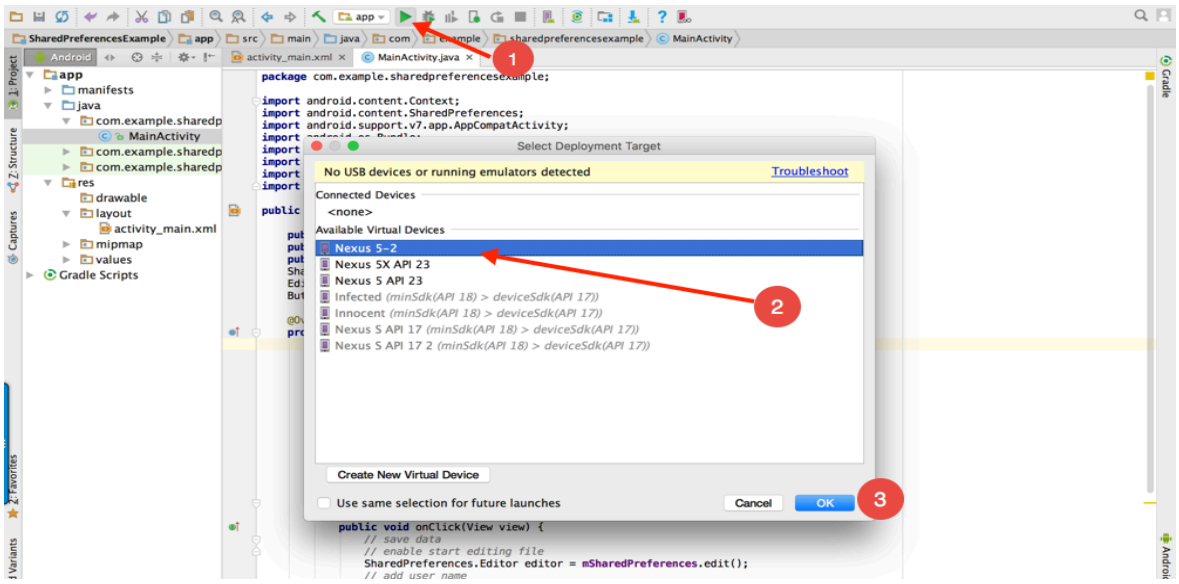
1. Upon creation, the onCreate method executes.
    a. Passes the Bundle named `savedInstanceState` to the superclass `AppCompatActivity`
    b. Initializes the variables declared in the first part

c. Creates a listener for the store button that upon click, will:
   i. Store the username and password the user gave as input in a SharedPreferences object
   ii. Commit the changes
   iii. Display a message confirming the successful save

Exploitation Instructions

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

1. Run the app. Enter your login credentials.



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

```
cd Library/Android/sdk/platform-tools
```

On Windows:

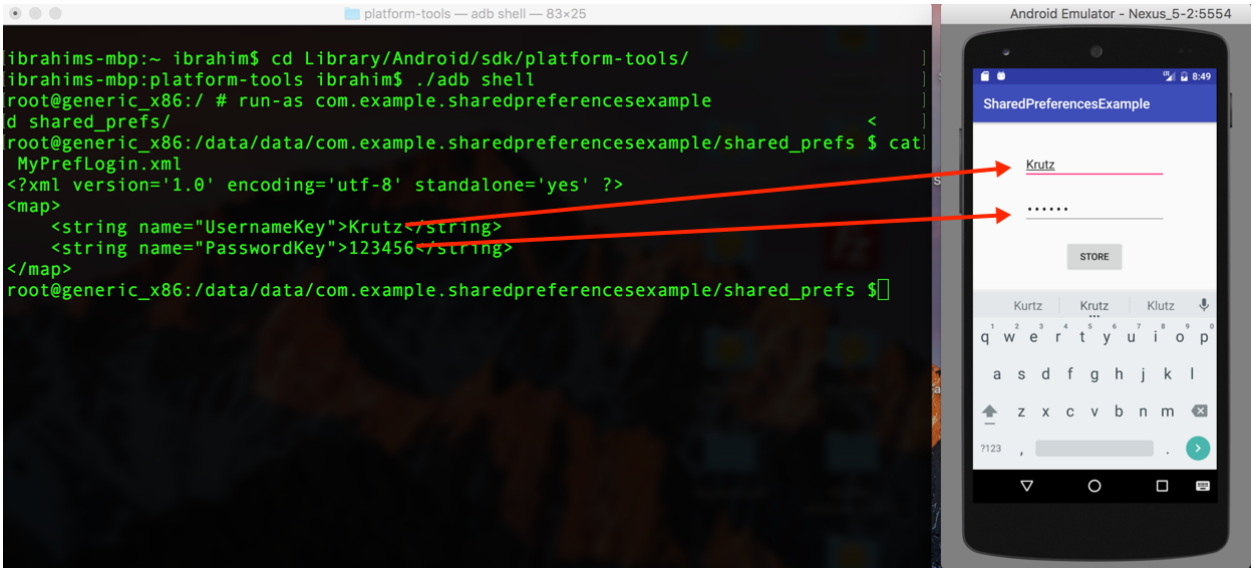
```
cd C:\Users\YOUR_USERNAME_HERE\AppData\Local\Android\sdk\platform-tools
```

   ii. 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
```

   iii. `run-as your_package_name`
   iv. `cd shared_prefs`
   v. `cat MyPrefLogin.xml`

3. Once you have executed the commands above, you will be able to read the data that was stored earlier.



Defense

Consider what kind of data is being stored. Is it sensitive? Does it need to be accessible to other apps? How is the data structured? Pick a location and format to store the data in based on what the answers to the questions are.

If the data is sensitive, you might want to consider storing it internally to take advantage of the simple access restrictions Android will provide. Assuming the access restrictions provided by internal storage do not work, we have two other options. We could a) encrypt the data on the phone using difficult-to-access keys, or b) store the data on a server elsewhere, pulling it down when needed through an encrypted connection. By a difficult-to-access key, we simply meant that these keys might be encrypted themselves, and will require user intervention in the form of a different password or something else.

In addition, we can use permissions to restrict access.

In general, if the app can access the data locally, a determined attacker will also be able to. However, we can make the effort not worth it. Remember that the financially-motivated attacker will not expend more time trying to get access to your data than the data itself is worth.

If you wish, we have created some example code you can insert into the app to see how encryption would make the login credentials hard to read. Keep in mind that the Caesar cipher, which we are using here, is very much broken in today's world. Do not use this code to secure data of any value.

```
package your_package_name;

import android.content.Context;
import android.content.SharedPreferences;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    public static final String MY_PREFERENCES = "MyPrefLogin" ;
    public static final String USERNAME = "UsernameKey" ;
    public static final String USER_PASSWORD = "PasswordKey" ;
    SharedPreferences mSharedPreferences;
    EditText mUsernameEditText, mPasswordEditText;
    Button mStoreButton;

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

        // initialize  the username EditText  instance
        mUsernameEditText = (EditText)findViewById(R.id.usernameEditText);

        // initialize  the password  EditText instance
        mPasswordEditText = (EditText)findViewById(R.id.passwordEditText);

        // initialize  the Button instance
        mStoreButton = (Button) findViewById(R.id.storeButton);

        // initialize SharedPreferences
        mSharedPreferences = getSharedPreferences(MY_PREFERENCES, Context.MODE_PRIVATE);

        // setting  on click listener for the button
        mStoreButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // Store data
                // enable start editing file
                SharedPreferences.Editor editor = mSharedPreferences.edit();
                // add user name
                editor.putString(USERNAME,
cipher(mUsernameEditText.getText().toString(),10));
                // add password
                editor.putString(USER_PASSWORD,cipher(
mPasswordEditText.getText().toString(),10));
                // store the update data
                editor.commit();
                //display message to inform the user that the stored
                Toast.makeText(MainActivity.this, "Your data has stored successfully!",
Toast.LENGTH_LONG).show();
            }
        });
    }
```

```
    // Cipher Method. Cipher encryption add shift for key
    String cipher(String msg, int shift){
        String s = "";
        int len = msg.length(); // get string length
        for(int x = 0; x < len; x++){
            char c = (char)(msg.charAt(x) + shift);  // shift every character
            s += c; // append the characters
        }
        return s;
    }
}
```

The entirety of `MainActivity.java` is posted above. The sections you'll want to change or add are highlighted. Running this and completing the same steps in `adb shell` will produce this: