Services

Background:

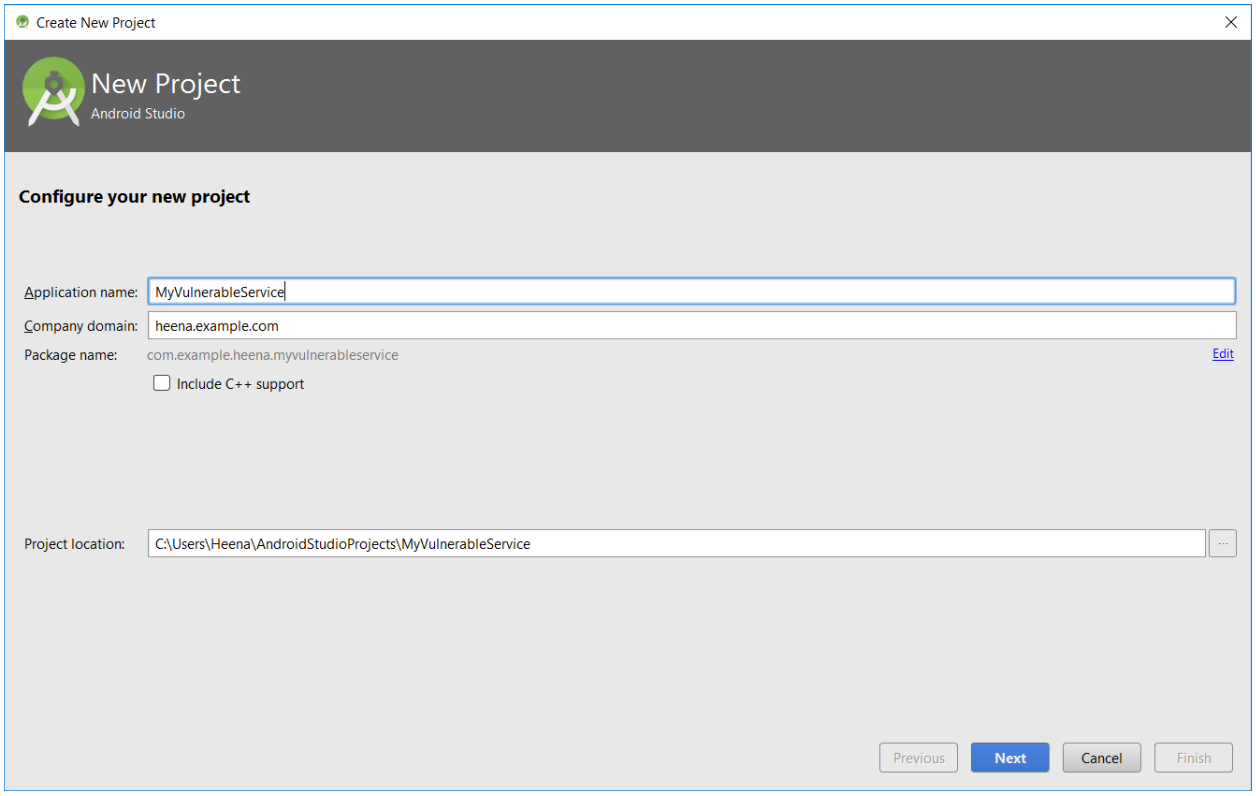
Services are one of 4 application components available within Android. They handle background processing and do not have a user interface. Operations that a service could perform include: playing music while the user switches to a different application or download apps from the internet without blocking the UI, allowing the user to continue performing other tasks. A service could be built to be local to an application and accessible to an Activity within that application or expose functionality to other applications over IPC. Services could be triggered when the system starts, or be triggered by an activity within the app. They could also be started when bound to by another app.

A service could be vulnerable to unauthorized access, if it is not secured through careful configuration. Android API Level 21 and above would not allow you to start a service using an implicit intent. They recommend avoiding the use of intent filters as well, to be certain which service responds to a call. Without an explicit intent, intent resolution could cause any service to start up which puts the user at risk. We'll see how explicit intents should be further secured to ensure that only apps we authorize can bind to our service.

The two samples we build as part of this walkthrough are examples of a vulnerable service - how an unauthorized app can bind to it to get data and a secure service. For simplicity, we'll have an app with an Activity and a Service and another with an activity that binds to the service. The Activity shall have Buttons to start and stop the service. We'll have toast notifications indicating that the buttons are indeed starting and stopping our service. We shall observe the Android Monitor to view results until we have a client app binding to the service to view data returned.

Steps to build the **Service**:

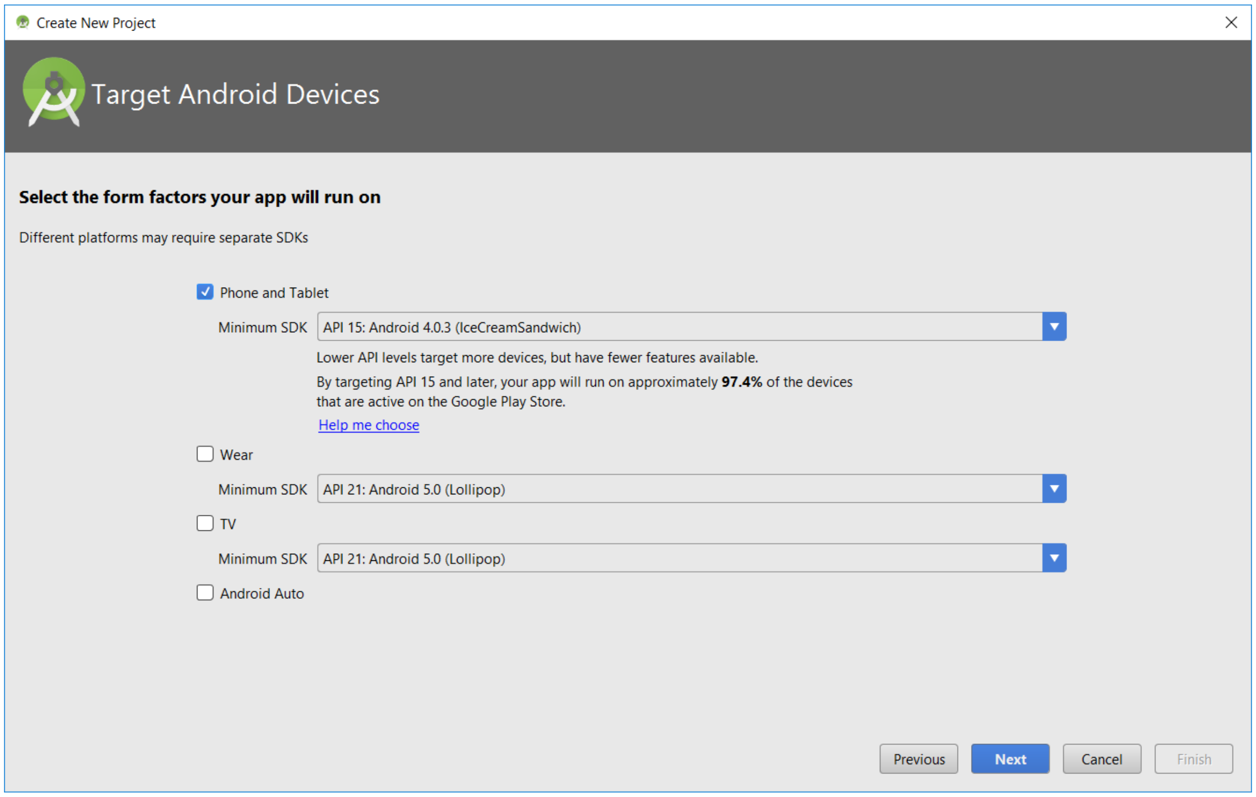
1. Open Android Studio and create new Android project and name it “MyVulnerableService”. Make sure to save the package name somewhere on your computer since you will need it in a later stage of this tutorial. Click next.



Configure your new project

Application name: MyVulnerableService
Company domain: heena.example.com
Package name: com.example.heena.myvulnerableservice
Include C++ support
Project location: C:\Users\Heena\AndroidStudioProjects\MyVulnerableService

2. Set the Minimum SDK to API Level 15 or whatever your default is, and then click Next.



Create New Project

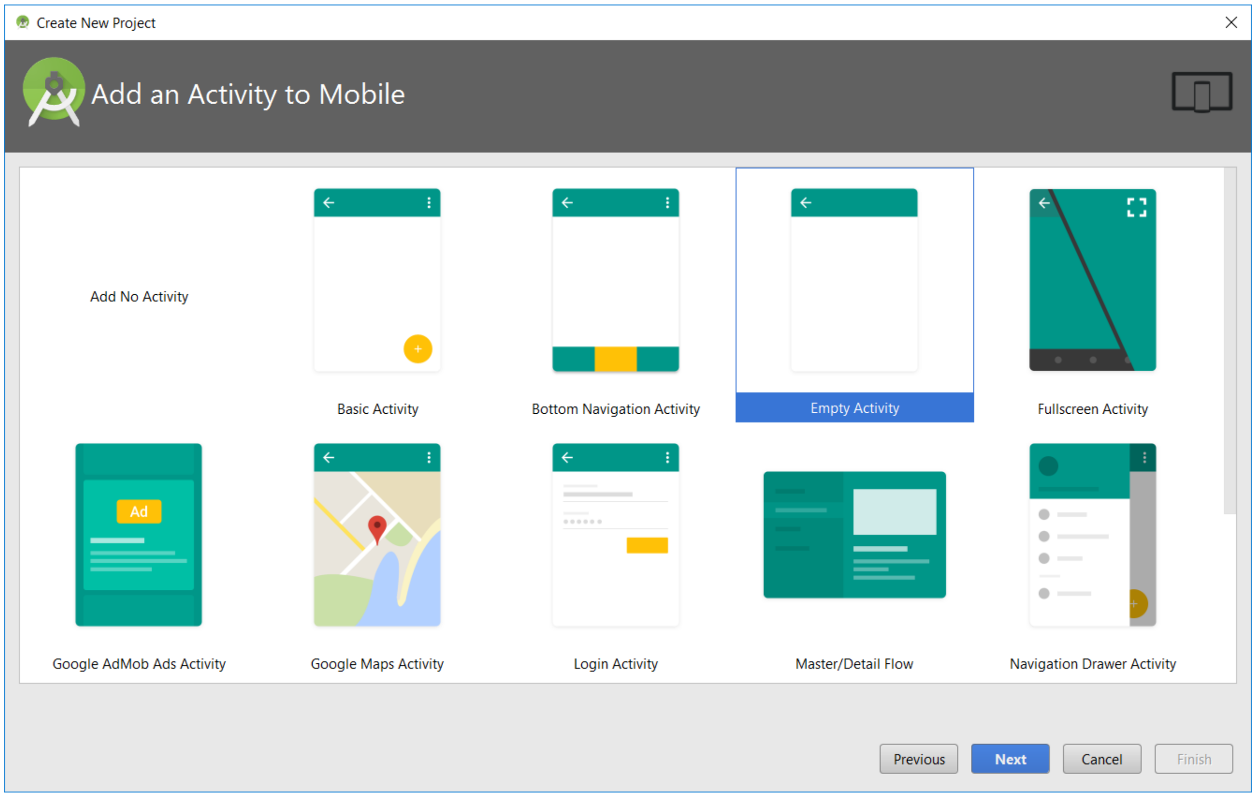
Target Android Devices

Select the form factors your app will run on

Different platforms may require separate SDKs

- [✓] Phone and Tablet
  - Minimum SDK: API 15: Android 4.0.3 (IceCreamSandwich)
  - Lower API levels target more devices, but have fewer features available.
  - By targeting API 15 and later, your app will run on approximately **97.4%** of the devices that are active on the Google Play Store.
  - Help me choose
- [ ] Wear
  - Minimum SDK: API 21: Android 5.0 (Lollipop)
- [ ] TV
  - Minimum SDK: API 21: Android 5.0 (Lollipop)
- [ ] Android Auto

3. Select Empty Activity, click on next and then finish to start the project



Create New Project

Add an Activity to Mobile

4. Open activity_main.xml which you'll find under app/res/layout. Delete the "Hello World" TextView.
5. Let's create an Activity that will allow the user to Start and Stop the service. Add 2 Buttons to a Relative Layout. We shall get to the details of these how we use them in our activity later in the tutorial. The code below adds the 2 Buttons to the layout.

```
    <?xml version="1.0" encoding="utf-8"?>
    <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:paddingTop="20dp"
        android:paddingBottom="20dp"
        android:paddingLeft="20dp"
        android:paddingRight="20dp"
        tools:context=" your.package.name.myvulnerableservice.MainActivity">

        <Button
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/start_service_btn"
            android:id="@+id/StartServiceBtn"
            />

        <Button
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/stop_service_btn"
            android:id="@+id/StopServiceBtn"
            android:layout_below="@+id/StartServiceBtn"
            android:layout_marginTop="10dp"/>

    </RelativeLayout>
```

6. The highlighted lines represent properties text and id that specify the text that goes on the button and the Id that identifies those buttons on the activity. The Id is created and assigned here. We shall see how to use the Id when we write code for our Activity. However, the text that goes on the buttons shouldn't be hard-coded in this file, but should be obtained from a different file that maintains all such strings. Add the following lines of code to the strings.xml file under res/values.

```
        <resources>
            <string name="app_name">MyVulnerableService</string>
            <string name="start_service_btn">Start Service</string>
            <string name="stop_service_btn">Stop Service</string>
        </resources>
```

7. Let's write a service that generates random numbers continuously once we start the service and continues to do so until we stop it. Under app/java, right click on the folder that contains MainActivity i.e. your.package.name.myvulnerableservice. Select New and then select Java Class. Give it the name "MyVulnerableService" and click OK. Leave the rest of the defaults as they are. Alternatively, you could create a new Service from the Menu which will extend Android's Service class for you and provide some you with some methods you need to override.

   Copy and paste the following code into your MyVulnerableService.java file. Make sure the package name at the top of the file is where your file lives.

```java
package your.package.name.myvulnerableservice;

import android.app.Service;
import android.content.Intent;
import android.os.Handler;
import android.os.IBinder;
import android.os.Message;
import android.os.Messenger;
import android.os.RemoteException;
import android.support.annotation.Nullable;
import android.util.Log;
import android.widget.Toast;


public class MyVulnerableService extends Service{

    private Messenger messenger=new Messenger(new RequestHandler());
    public static final int REQUEST_COUNT = 0;
    public boolean isServiceRunning=false;
    private int random = 0 ;

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        isServiceRunning = true;
        Thread service_thread = new Thread(new Runnable() {
            @Override
            public void run() {
                startRandomNumGenerationService();
            }
        });
        service_thread.start();
        return START_STICKY;
    }

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return messenger.getBinder();
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        stopRandomNumGenerationService();
    }

    private static class RequestHandler extends Handler {
        @Override
        public void handleMessage(Message message){
            switch (message.what){
                case REQUEST_COUNT:
                    Message message_to_send = Message.obtain(null,REQUEST_COUNT);
                    message_to_send.arg1 = getRandomNumber();
                    try{
                        message.replyTo.send(message_to_send);
                    }catch (RemoteException e){
                        Log.i("My Service",e.getMessage());
                    }
            }

            super.handleMessage(message);
        }
    }

    private void startRandomNumGenerationService(){
        while(isServiceRunning){
            try{
                Thread.sleep(1000);
                if(isServiceRunning){
                    random = getRandomNumber();
                    Log.i("My Math Service","New Random Number is :"+ random);
                }
            }catch (InterruptedException e){
                Log.i("My Math Service","Service Thread Interrupted");
            }
        }
    }

    private void stopRandomNumGenerationService(){
        isServiceRunning = false;
        Toast.makeText(getApplicationContext(),"Math Service Stopped",
Toast.LENGTH_SHORT).show();
    }

    public static int getRandomNumber() {
        return (int)(Math.random()*100+1);
    }

}
```

8. Let’s add this service to our manifest file. This being a bound service, we want to have another app we create communicate with this service. It is for this reason, we shall have the android:exported attribute of the service set to true. This is usually accompanied by Intent filters within the service declaration but you can no longer have implicit intents trigger services. Doing so will trigger an exception. This was added as part of Android 5.0 (API level

21), as a security feature to prevent other apps from accessing this service by using an implicit intent. However, constructing an explicit intent is not too difficult either. This allows other apps to gain access to service. We'll get to one such dangerous activity later in this tutorial.

For now, copy and paste the following code into the AndroidManifest.xml file under app/manifests.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package=" your.package.name.myvulnerableservice">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <service
            android:name=".MyVulnerableService"
            android:exported="true">
        </service>
    </application>
</manifest>
```

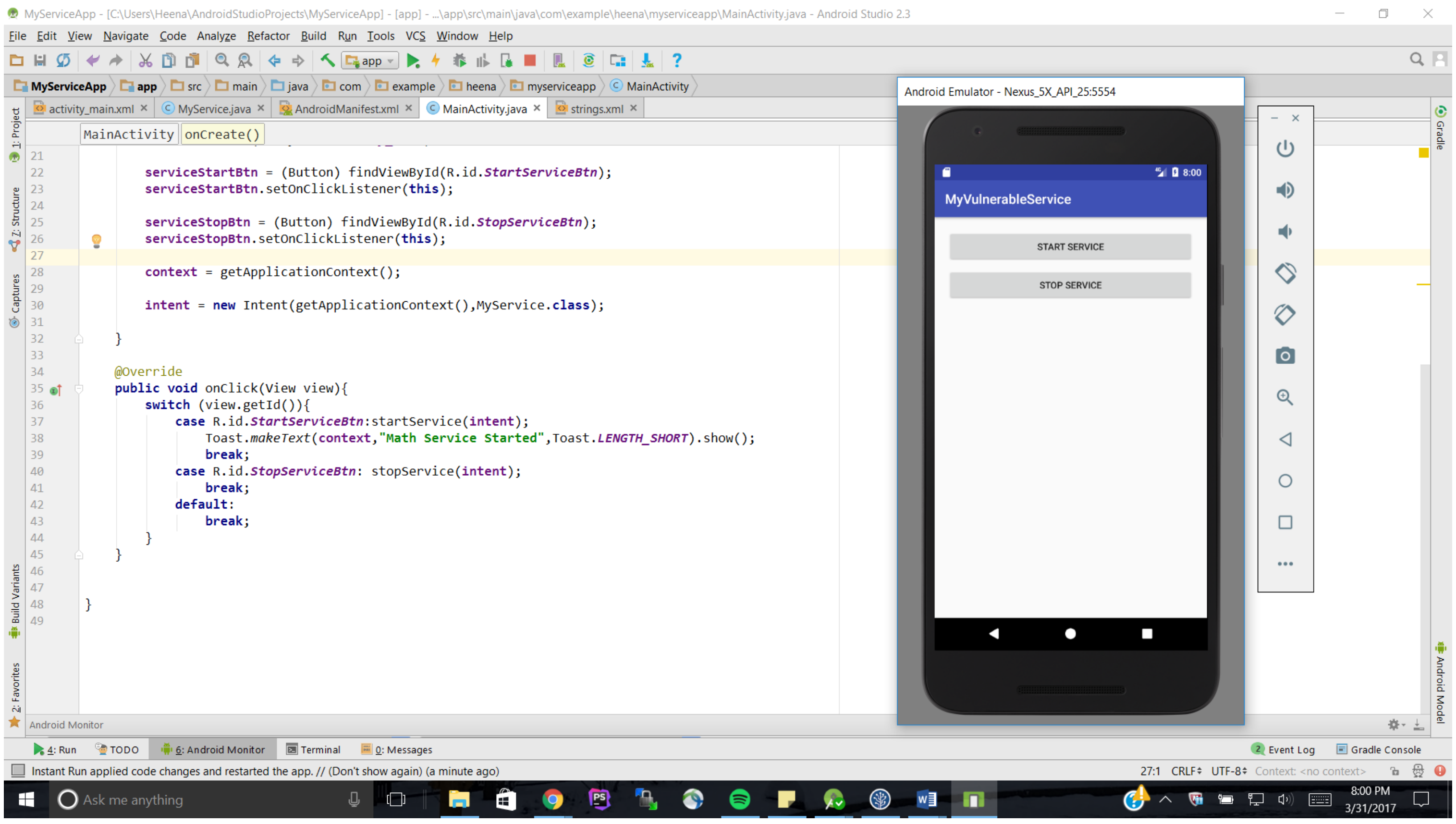
You only need to add the highlighted portion to your manifest file. The rest should already be in there, since your activity is added to the manifest by default.

9. Let's go ahead and write the code for our Activity. This will allow us to start and stop the service. Copy and paste the following code in your MainActivity.java file.

```
package your.package.name.myvulnerableservice;

import android.content.Context;
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.Toast;

public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    private Button serviceStartBtn, serviceStopBtn;
    private Context context;
    private Intent intent;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        serviceStartBtn = (Button) findViewById(R.id.StartServiceBtn);
        serviceStartBtn.setOnClickListener(this);

        serviceStopBtn = (Button) findViewById(R.id.StopServiceBtn);
        serviceStopBtn.setOnClickListener(this);

        context = getApplicationContext();

        intent = new Intent(getApplicationContext(),MyVulnerableService.class);

    }

    @Override
    public void onClick(View view){
        switch (view.getId()){
            case R.id.StartServiceBtn:startService(intent);
                Toast.makeText(context,"Math Service
Started",Toast.LENGTH_SHORT).show();
                break;
            case R.id.StopServiceBtn: stopService(intent);
                break;
            default:
                break;
        }
    }
}
```

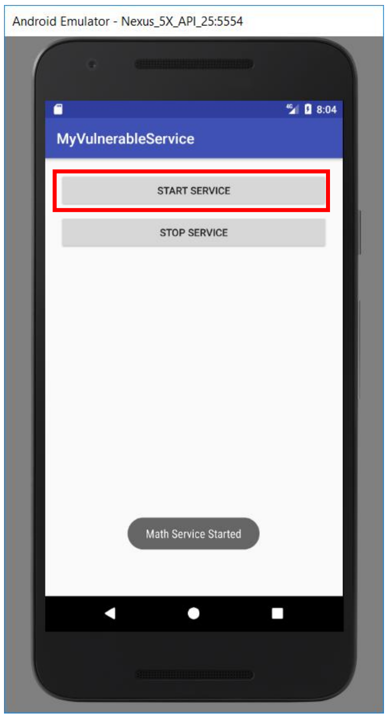
The highlighted section shows you how we're accessing the Buttons on our Activity by id. And how they're used to start and stop the service.

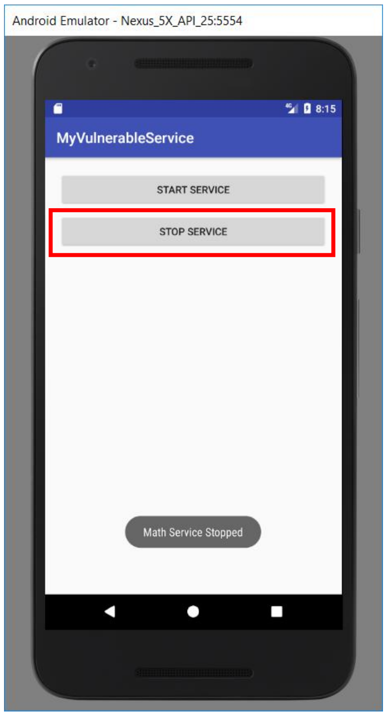
10. Let's test our service. Click on the run icon on the toolbar. You should have the emulator show up in a minute with the Activity and the start and stop service buttons.


MyServiceApp - [C:\Users\Heena\AndroidStudioProjects\MyServiceApp] - [app] - ...\app\src\main\java\com\example\heena\myserviceapp\MainActivity.java - Android Studio 2.3
Android Emulator - Nexus_5X_API_25:5554
MyVulnerableService
START SERVICE
STOP SERVICE
Instant Run applied code changes and restarted the app. // (Don't show again) (a minute ago)

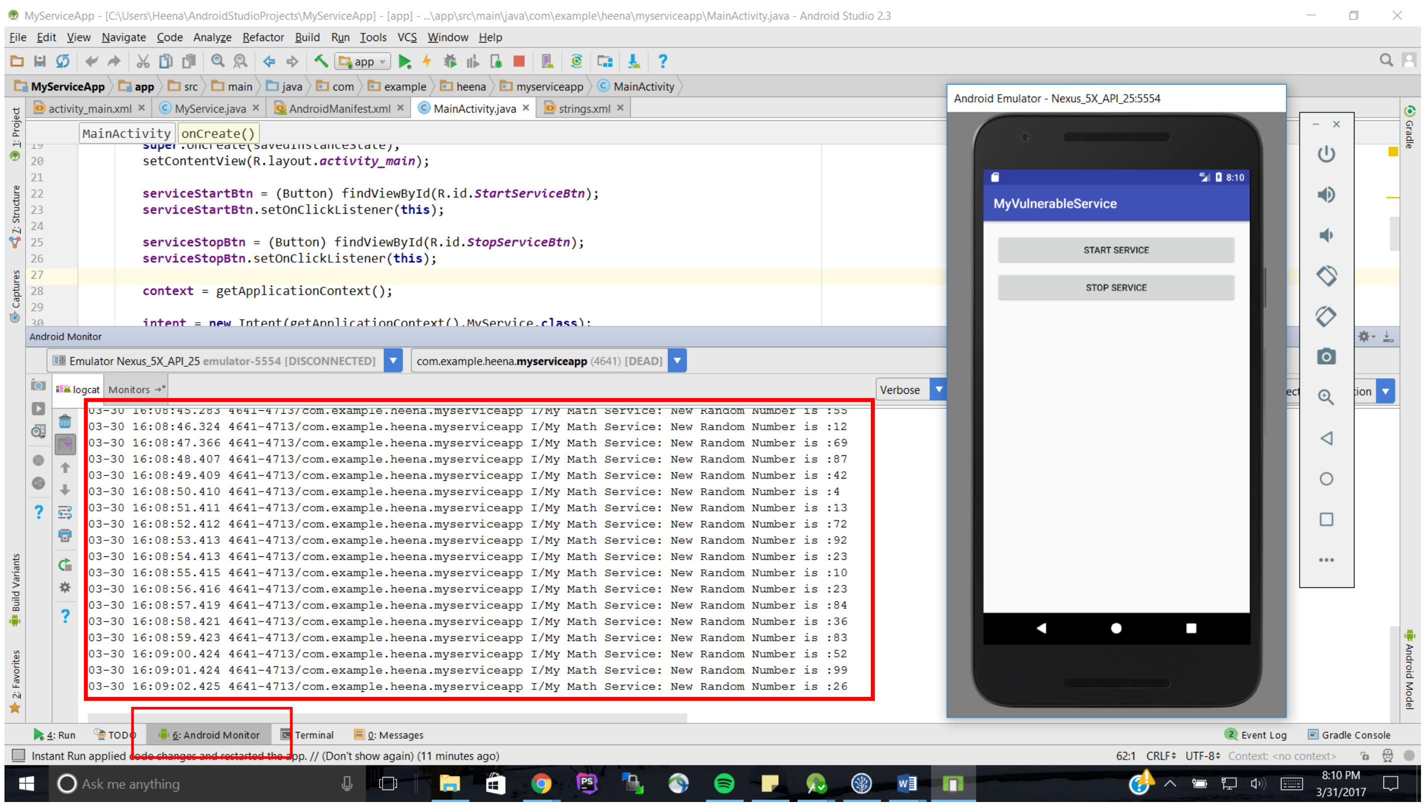
```
        serviceStartBtn = (Button) findViewById(R.id.StartServiceBtn);
        serviceStartBtn.setOnClickListener(this);

        serviceStopBtn = (Button) findViewById(R.id.StopServiceBtn);
        serviceStopBtn.setOnClickListener(this);

        context = getApplicationContext();

        intent = new Intent(getApplicationContext(),MyService.class);

    }

    @Override
    public void onClick(View view){
        switch (view.getId()){
            case R.id.StartServiceBtn:startService(intent);
                Toast.makeText(context,"Math Service Started",Toast.LENGTH_SHORT).show();
                break;
            case R.id.StopServiceBtn: stopService(intent);
                break;
            default:
                break;
        }
    }


}
```

Click on the Start Service Button – a Toast should show up indicating the service has started running. Stopping the service will also show a similar toast.





Observe the Android Monitor and you should see the service running, displaying random numbers.


MyServiceApp - [C:\Users\Heena\AndroidStudioProjects\MyServiceApp] - [app] - ...\app\src\main\java\com\example\heena\myserviceapp\MainActivity.java - Android Studio 2.3
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
MyServiceApp app src main java com example heena myserviceapp MainActivity
activity_main.xml MyService.java AndroidManifest.xml MainActivity.java strings.xml
MainActivity onCreate()
Android Monitor
Emulator Nexus_5X_API_25 emulator-5554 [DISCONNECTED]
com.example.heena.myserviceapp (4641) [DEAD]
logcat Monitors
Verbose
4: Run TODO 6: Android Monitor Terminal 0: Messages
Event Log Gradle Console
Android Emulator - Nexus_5X_API_25:5554
MyVulnerableService
START SERVICE
STOP SERVICE

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

        serviceStartBtn = (Button) findViewById(R.id.StartServiceBtn);
        serviceStartBtn.setOnClickListener(this);

        serviceStopBtn = (Button) findViewById(R.id.StopServiceBtn);
        serviceStopBtn.setOnClickListener(this);

        context = getApplicationContext();

        intent = new Intent(getApplicationContext(),MyService.class);
```

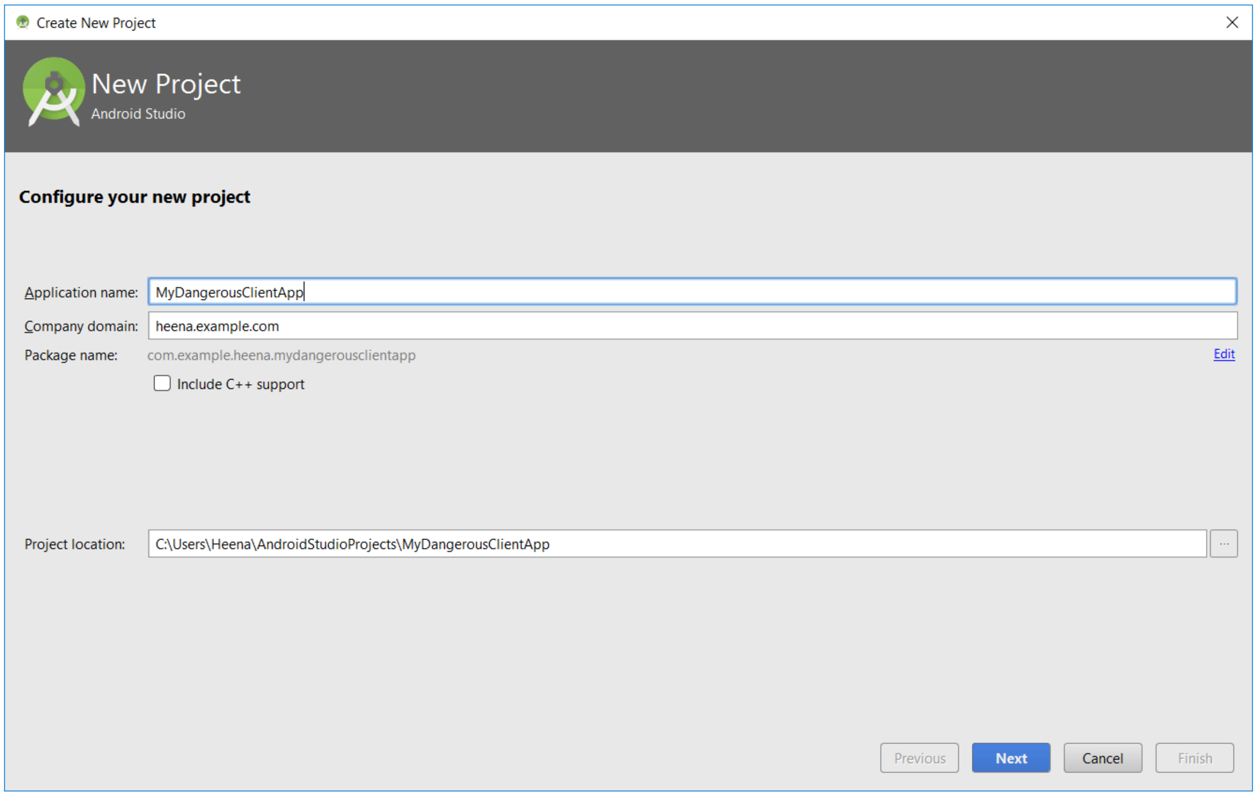
```
03-30 16:08:45.283 4641-4713/com.example.heena.myserviceapp I/My Math Service: New Random Number is :55
03-30 16:08:46.324 4641-4713/com.example.heena.myserviceapp I/My Math Service: New Random Number is :12
03-30 16:08:47.366 4641-4713/com.example.heena.myserviceapp I/My Math Service: New Random Number is :69
03-30 16:08:48.407 4641-4713/com.example.heena.myserviceapp I/My Math Service: New Random Number is :87
03-30 16:08:49.409 4641-4713/com.example.heena.myserviceapp I/My Math Service: New Random Number is :42
03-30 16:08:50.410 4641-4713/com.example.heena.myserviceapp I/My Math Service: New Random Number is :4
03-30 16:08:51.411 4641-4713/com.example.heena.myserviceapp I/My Math Service: New Random Number is :13
03-30 16:08:52.412 4641-4713/com.example.heena.myserviceapp I/My Math Service: New Random Number is :72
03-30 16:08:53.413 4641-4713/com.example.heena.myserviceapp I/My Math Service: New Random Number is :92
03-30 16:08:54.413 4641-4713/com.example.heena.myserviceapp I/My Math Service: New Random Number is :23
03-30 16:08:55.415 4641-4713/com.example.heena.myserviceapp I/My Math Service: New Random Number is :10
03-30 16:08:56.416 4641-4713/com.example.heena.myserviceapp I/My Math Service: New Random Number is :23
03-30 16:08:57.419 4641-4713/com.example.heena.myserviceapp I/My Math Service: New Random Number is :84
03-30 16:08:58.421 4641-4713/com.example.heena.myserviceapp I/My Math Service: New Random Number is :36
03-30 16:08:59.423 4641-4713/com.example.heena.myserviceapp I/My Math Service: New Random Number is :83
03-30 16:09:00.424 4641-4713/com.example.heena.myserviceapp I/My Math Service: New Random Number is :52
03-30 16:09:01.424 4641-4713/com.example.heena.myserviceapp I/My Math Service: New Random Number is :99
03-30 16:09:02.425 4641-4713/com.example.heena.myserviceapp I/My Math Service: New Random Number is :26
```

Instant Run applied code changes and restarted the app. // (Don't show again) (11 minutes ago)

Let's build the client app that will bind to the service we just created and try and get data back from the service.

Steps to build the **Client app**:

1. Open Android Studio and create new Android project and name it "MyDangerousClientApp". Make sure to save the package name somewhere on your computer since you will need it in a later stage of this tutorial. Click next.


Create New Project

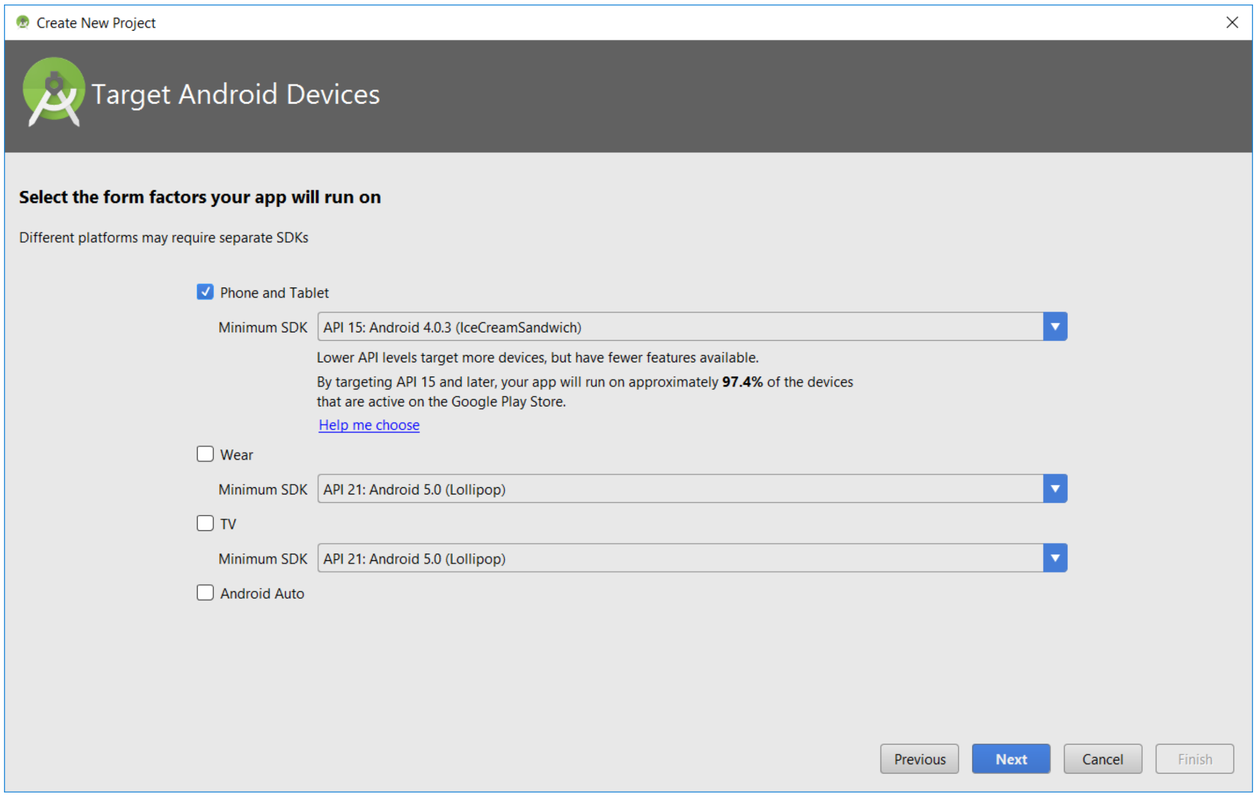
New Project
Android Studio

Configure your new project

Application name: MyDangerousClientApp
Company domain: heena.example.com
Package name: com.example.heena.mydangerousclientapp Edit
☐ Include C++ support

Project location: C:\Users\Heena\AndroidStudioProjects\MyDangerousClientApp

2. Set the Minimum SDK to API Level 15 or whatever your default is, and then click Next.



Create New Project

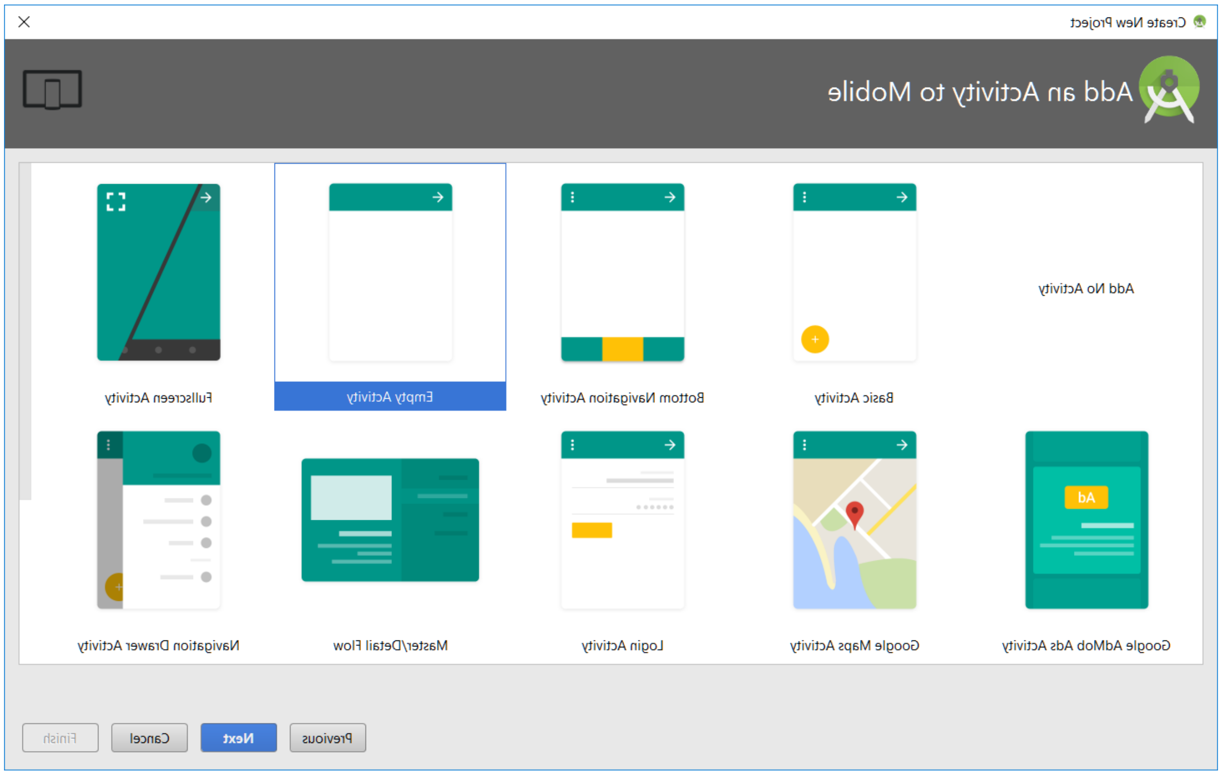
Target Android Devices

Select the form factors your app will run on

Different platforms may require separate SDKs

- [x] Phone and Tablet
  - Minimum SDK: API 15: Android 4.0.3 (IceCreamSandwich)
  - Lower API levels target more devices, but have fewer features available.
  - By targeting API 15 and later, your app will run on approximately **97.4%** of the devices that are active on the Google Play Store.
  - Help me choose
- [ ] Wear
  - Minimum SDK: API 21: Android 5.0 (Lollipop)
- [ ] TV
  - Minimum SDK: API 21: Android 5.0 (Lollipop)
- [ ] Android Auto

3. Select Empty Activity, click on next and then finish to start the project



Create New Project

Add an Activity to Mobile

4. Open activity_main.xml which you'll find under app/res/layout. Delete the "Hello World" TextView.
5. Let's create an Activity that will allow the user to Bind to the service and Start the service. Add 2 Buttons to a Relative Layout just like we did for the service. We'll also add a textView that will display the random number returned by the service to this app. We shall get to the details of these how we use these buttons in our activity later in the tutorial. The code below adds these 2 Buttons and the TextView to the layout.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingTop="20dp"
    android:paddingBottom="20dp"
    android:paddingLeft="20dp"
    android:paddingRight="20dp"
    tools:context="your.package.name.MainActivity">

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/bind_to_service_btn"
        android:id="@+id/BindToServiceBtn"
        />

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/invoke_service_btn"
        android:id="@+id/InvokeServiceBtn"
        android:layout_below="@+id/BindToServiceBtn"
        android:layout_marginTop="10dp"/>


    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_below="@+id/InvokeServiceBtn"
        android:textSize = "20sp"
        android:layout_marginTop="120dp"
        android:id="@+id/ResultView"/>

</RelativeLayout>
```

6. Let's add the strings that represents text that goes on the buttons. Copy and paste the following lines of code to the strings.xml file under res/values.

```
        <resources>
            <string name="app_name">MyDangerousClientApp</string>
            <string name="invoke_service_btn">Get New Random Number</string>
            <string name="bind_to_service_btn">Bind to Math Service</string>
        </resources>
```

7. Let's add the activity to the manifest file. Copy and paste the following lines of code to the AndroidManifest.xml file.

```
        <?xml version="1.0" encoding="utf-8"?>
        <manifest xmlns:android="http://schemas.android.com/apk/res/android"
            package="your.package.name.mydangerousclientapp">

            <application
                android:allowBackup="true"
                android:icon="@mipmap/ic_launcher"
                android:label="@string/app_name"
                android:roundIcon="@mipmap/ic_launcher_round"
                android:supportsRtl="true"
                android:theme="@style/AppTheme">
                <activity android:name=".MainActivity">
                    <intent-filter>
                        <action android:name="android.intent.action.MAIN" />
                        <category android:name="android.intent.category.LAUNCHER" />
                    </intent-filter>
                </activity>
            </application>
            </manifest>
```

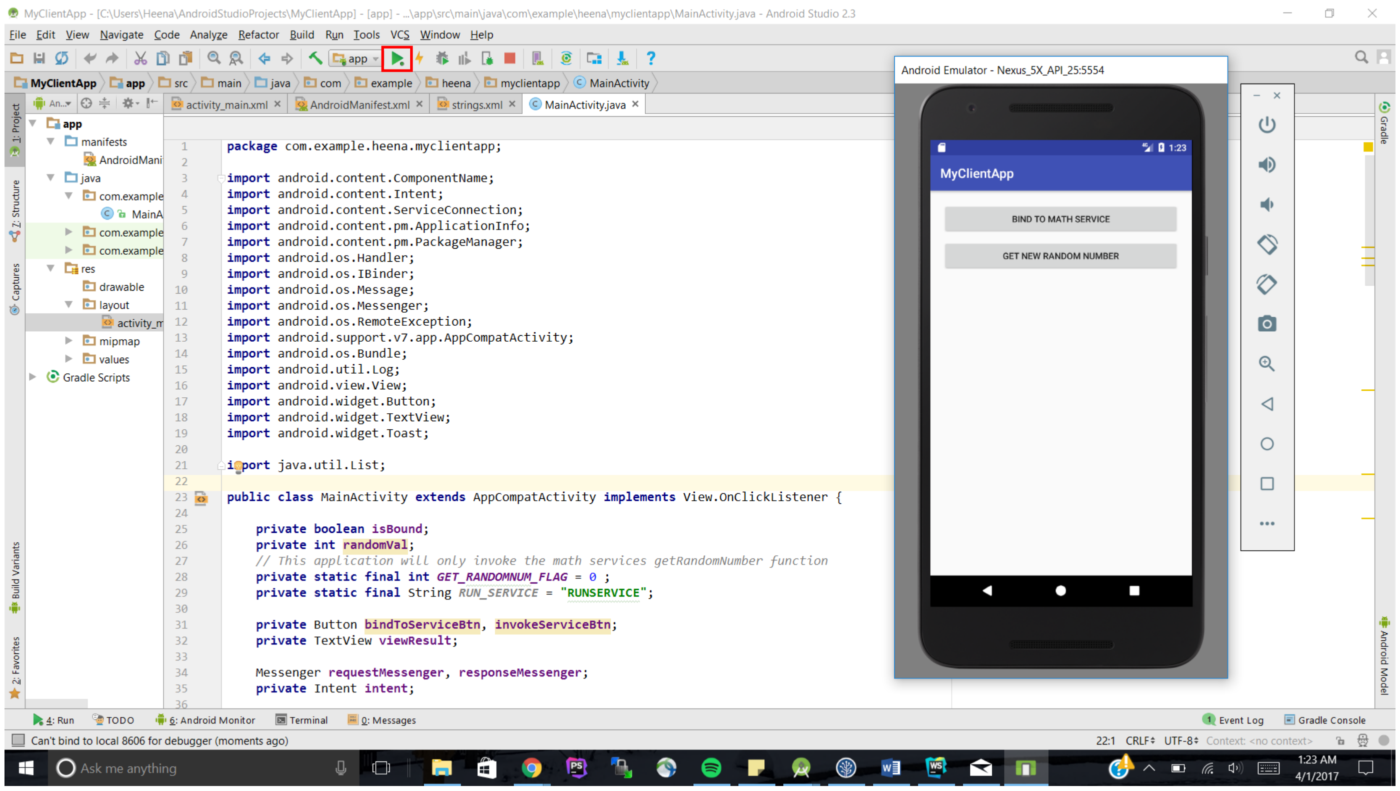
8. We know that we cannot access a service without having an explicit intent. Creating an explicit intent requires the package name and the app name that contains the service. Any external app doesn't generally know the package name of the service. However, there are many ways of determining the package name given the app name in code. There are also apps on the Play Store that can give you the package names of all apps on your device ! This makes it trivial to create an explicit intent and bind to the service. (assuming the class name of the service is same as that of the app name here)

   Now since our service has its exported attribute set to true, any app can create the explicit intent from the package name and start the service.This is how our dangerous app works!

   Copy and paste the following code into the MainActivity.java file.

```
package your.package.name.mydangerousclientapp;

import android.content.ComponentName;
import android.content.Intent;
import android.content.ServiceConnection;
import android.content.pm.ApplicationInfo;
import android.content.pm.PackageManager;
import android.os.Handler;
import android.os.IBinder;
import android.os.Message;
import android.os.Messenger;
import android.os.RemoteException;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import java.util.List;

public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    private boolean isBound;
    private int randomVal;
    // This application will only invoke the math service’s getRandomNumber()
function
    private static final int GET_RANDOMNUM_FLAG = 0 ;
    private static final String RUN_SERVICE = "RUNSERVICE";

    private Button bindToServiceBtn, invokeServiceBtn;
    private TextView viewResult;

    Messenger requestMessenger, responseMessenger;
    private Intent intent;


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

    viewResult = (TextView)findViewById(R.id.ResultView);
    bindToServiceBtn = (Button)findViewById(R.id.BindToServiceBtn);
    bindToServiceBtn.setOnClickListener(this);
    invokeServiceBtn = (Button)findViewById(R.id.InvokeServiceBtn);
    invokeServiceBtn.setOnClickListener(this);


    final PackageManager pm = getPackageManager();

    //get a list of installed apps.
    List<ApplicationInfo> applications =
pm.getInstalledApplications(PackageManager.GET_META_DATA);

    for (ApplicationInfo applicationInfo: applications) {

        CharSequence appName = pm.getApplicationLabel(applicationInfo);

        if((appName).equals("MyVulnerableService"))
        {
            intent = new Intent();
            intent.setComponent(new
ComponentName(applicationInfo.packageName,applicationInfo.packageName +
"."+appName));
            break;
        }
        else{
            continue;
        }
    }

}

    ServiceConnection serviceConnection = new ServiceConnection() {
        @Override
        public void onServiceConnected(ComponentName arg0, IBinder binder) {
            //wrap the binder returned from the onBind method in the Service
            requestMessenger=new Messenger(binder);
            /*wrap the Handler that will handle the response being sent from the
service*/
            responseMessenger=new Messenger(new ResponseHandler());
            isBound=true;
        }
        @Override
        public void onServiceDisconnected(ComponentName arg0) {
            requestMessenger=null;
            responseMessenger=null;
            isBound = false;
        }
    };

    private class ResponseHandler extends Handler{

        @Override
        public void handleMessage(Message message) {
            randomVal = 0;
            /*message sent from the messenger
            that holds an instance of the Binder
             returned from the onBind Method of the service*/

            switch (message.what){
                case GET_RANDOMNUM_FLAG:
                    randomVal = message.arg1;
                    viewResult.setText("Math Service - Random number is " +
randomVal);
                    break;
                default:
                    break;

            }
            super.handleMessage(message);
        }
    }


    private void bindToService(){
        //bind to the service using the explicit intent we created
        bindService(intent,serviceConnection,BIND_AUTO_CREATE);
        Toast.makeText(getApplicationContext(),"Bound to Math
Service",Toast.LENGTH_SHORT).show();
    }
    private void fetchNewRandomNum(){
        if(isBound){
            Message request = Message.obtain(null,GET_RANDOMNUM_FLAG);
            /*Message's replyTo attribute takes a messenger indicating
            which messenger the reply should be sent to
             once we have received something from the service*/
            try{
                //set receiving messenger
                request.replyTo = responseMessenger;
                //send request to service
                requestMessenger.send(request);
            }catch (RemoteException e){
                Log.i("My Dangerous Client App",e.getMessage());
            }

        }
    }

    @Override
    public void onClick(View view){
        switch (view.getId()){
            case R.id.BindToServiceBtn: bindToService();
                break;
            case R.id.InvokeServiceBtn: fetchNewRandomNum();
                break;
            default:break;
        }
    }
}
```

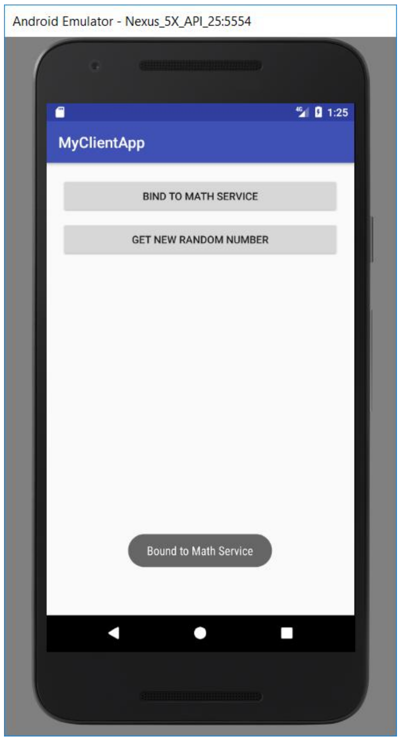
9. Let’s test our dangerous app. We first start the service app like we did earlier in this tutorial. We then start the “dangerous” client app, bind to the service and run it. Since our service is exported without any restrictions to enable access any app can do what our dangerous app does. Run the app. You should see the activity display the 2 buttons we added

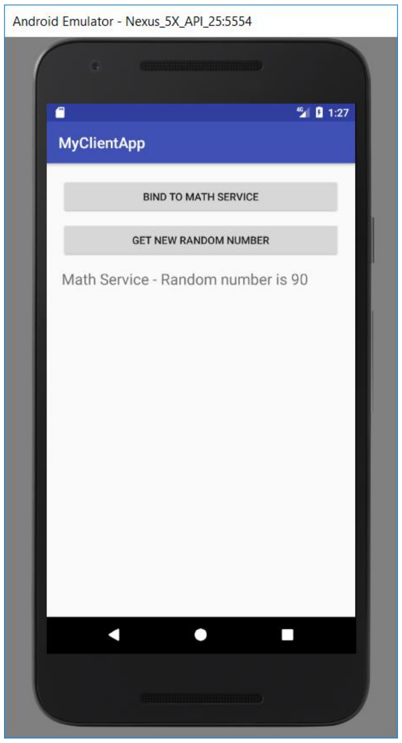

MyClientApp - [C:\Users\Heena\AndroidStudioProjects\MyClientApp] - [app] - ...\app\src\main\java\com\example\heena\myclientapp\MainActivity.java - Android Studio 2.3
Android Emulator - Nexus_5X_API_25:5554
MyClientApp
BIND TO MATH SERVICE
GET NEW RANDOM NUMBER

```java
package com.example.heena.myclientapp;

import android.content.ComponentName;
import android.content.Intent;
import android.content.ServiceConnection;
import android.content.pm.ApplicationInfo;
import android.content.pm.PackageManager;
import android.os.Handler;
import android.os.IBinder;
import android.os.Message;
import android.os.Messenger;
import android.os.RemoteException;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import java.util.List;

public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    private boolean isBound;
    private int randomVal;
    // This application will only invoke the math services getRandomNumber function
    private static final int GET_RANDOMNUM_FLAG = 0 ;
    private static final String RUN_SERVICE = "RUNSERVICE";

    private Button bindToServiceBtn, invokeServiceBtn;
    private TextView viewResult;

    Messenger requestMessenger, responseMessenger;
    private Intent intent;
```

```
Can't bind to local 8606 for debugger (moments ago)
```

Then bind to the service before getting a new random number.





Math Service - Random number is 90

Let's secure our service so that only an app that we'd like to give access to, could bind to our service. We can do these using intents. We add additional parameters to our explicit intent, that we use to bind to the service. On the service end, we check if have received the parameters we expect to find. This ensures that despite having the service's android:exported attribute set to true, we do not expose our service to any other app that may construct an explicit intent.

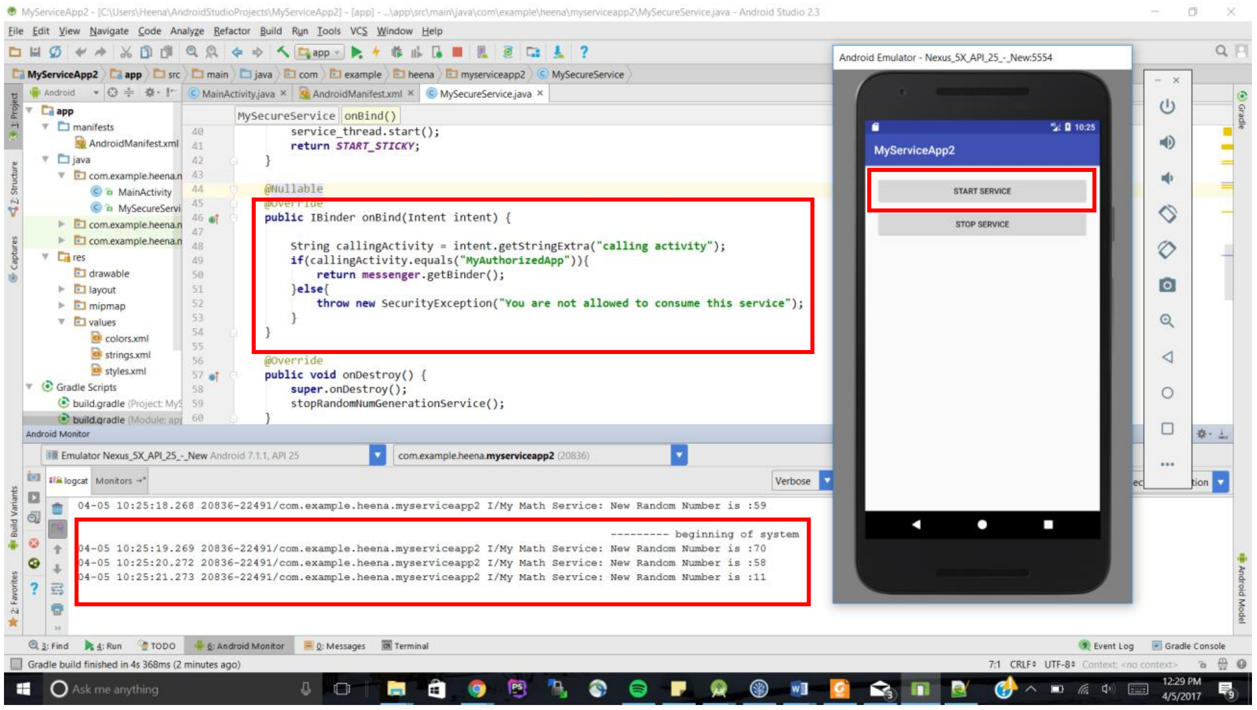
10. Make the following changes to the onCreate() method of the MainActivity.java file in the Dangerous Client App. Alternately you could create a copy of the Client app and Service apps, and make changes to those apps.

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

            viewResult = (TextView)findViewById(R.id.ResultView);
            bindToServiceBtn = (Button)findViewById(R.id.BindToServiceBtn);
            bindToServiceBtn.setOnClickListener(this);
            invokeServiceBtn = (Button)findViewById(R.id.InvokeServiceBtn);
            invokeServiceBtn.setOnClickListener(this);

            intent = new Intent();
            //we now know both the package and the name of the service's class
        intent.setComponent(new
        ComponentName("your.package.name.myvulnerableservice","your.package.name.
        myvulnerableservice.MyVulnerableService"));
        intent.putExtra("calling activity","MyAuthorizedApp");
        }
```

11. Let's change the Service to accept requests to bind only from this authorized app. Make the following changes to onBind() method of the MyVulnerableService.java file in the MyVulnerableServiceApp project.

```
@Nullable
@Override
public IBinder onBind(Intent intent) {

    String callingActivity = intent.getStringExtra("calling activity");
    if(callingActivity != null && callingActivity.equals("MyAuthorizedApp")){
                //we return the binder to a client we authorize
                return messenger.getBinder();
            }else{
                throw new SecurityException("You are not allowed to consume this
        service");
            }
        }
```

On testing an app that tries binding to this service with just the explicit intent, this is what you shall see. Start the service that has the updated code in the onBind() method.

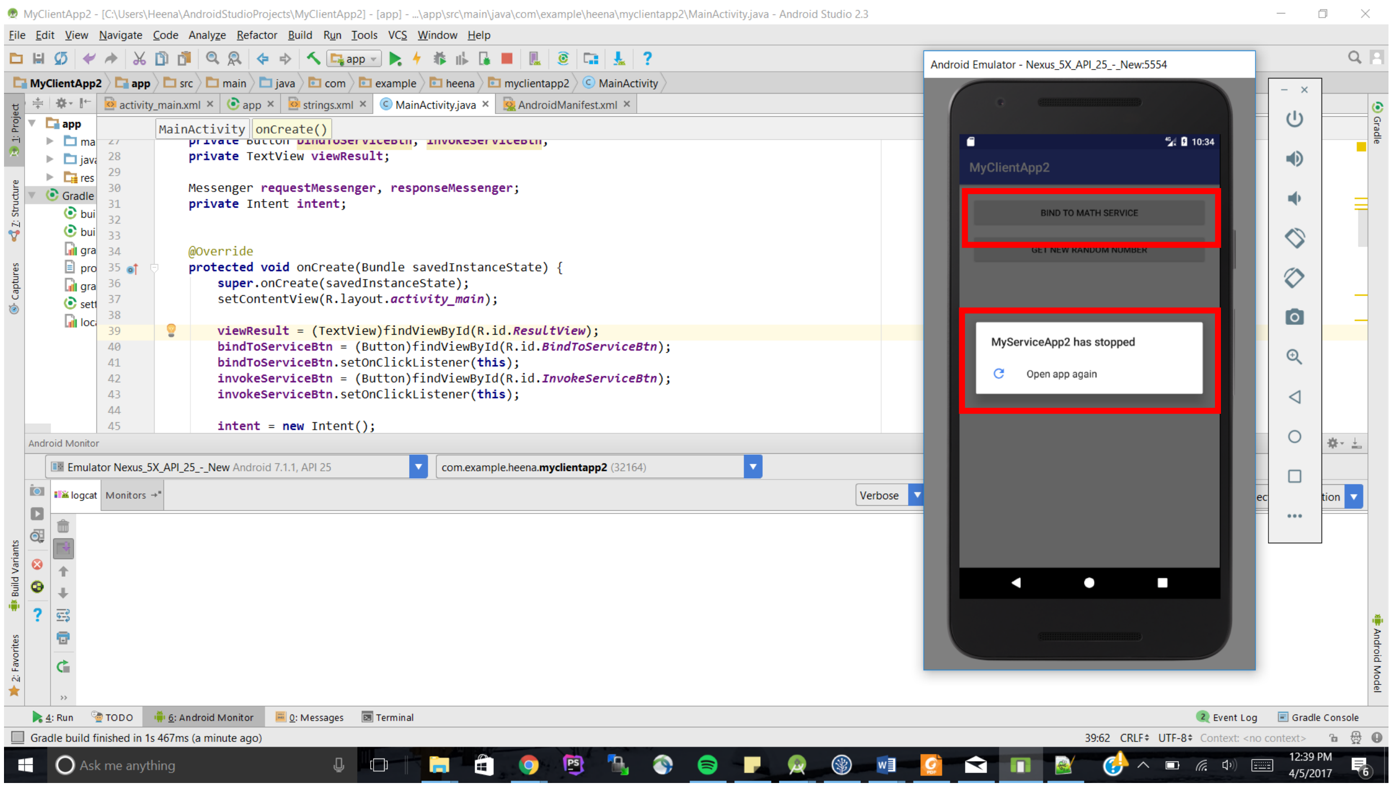


```
        service_thread.start();
        return START_STICKY;
    }

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {

        String callingActivity = intent.getStringExtra("calling activity");
        if(callingActivity.equals("MyAuthorizedApp")){
            return messenger.getBinder();
        }else{
            throw new SecurityException("You are not allowed to consume this service");
        }
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        stopRandomNumGenerationService();
    }
```

```
04-05 10:25:18.268 20836-22491/com.example.heena.myserviceapp2 I/My Math Service: New Random Number is :59

                                                                    --------- beginning of system
04-05 10:25:19.269 20836-22491/com.example.heena.myserviceapp2 I/My Math Service: New Random Number is :70
04-05 10:25:20.272 20836-22491/com.example.heena.myserviceapp2 I/My Math Service: New Random Number is :58
04-05 10:25:21.273 20836-22491/com.example.heena.myserviceapp2 I/My Math Service: New Random Number is :11
```

MyServiceApp2
START SERVICE
STOP SERVICE
Android Emulator - Nexus_5X_API_25_-_New:5554
Emulator Nexus_5X_API_25_-_New Android 7.1.1, API 25
com.example.heena.myserviceapp2 (20836)
Gradle build finished in 4s 368ms (2 minutes ago)

Running the client app that has the explicit intent (without additional parameters). Trying to bind to the service will cause the service app to crash and throw an exception.



MyClientApp2 - [C:\Users\Heena\AndroidStudioProjects\MyClientApp2] - [app] - ...\app\src\main\java\com\example\heena\myclientapp2\MainActivity.java - Android Studio 2.3

```java
    private Button bindToServiceBtn, invokeServiceBtn;
    private TextView viewResult;

    Messenger requestMessenger, responseMessenger;
    private Intent intent;


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

        viewResult = (TextView)findViewById(R.id.ResultView);
        bindToServiceBtn = (Button)findViewById(R.id.BindToServiceBtn);
        bindToServiceBtn.setOnClickListener(this);
        invokeServiceBtn = (Button)findViewById(R.id.InvokeServiceBtn);
        invokeServiceBtn.setOnClickListener(this);

        intent = new Intent();
```

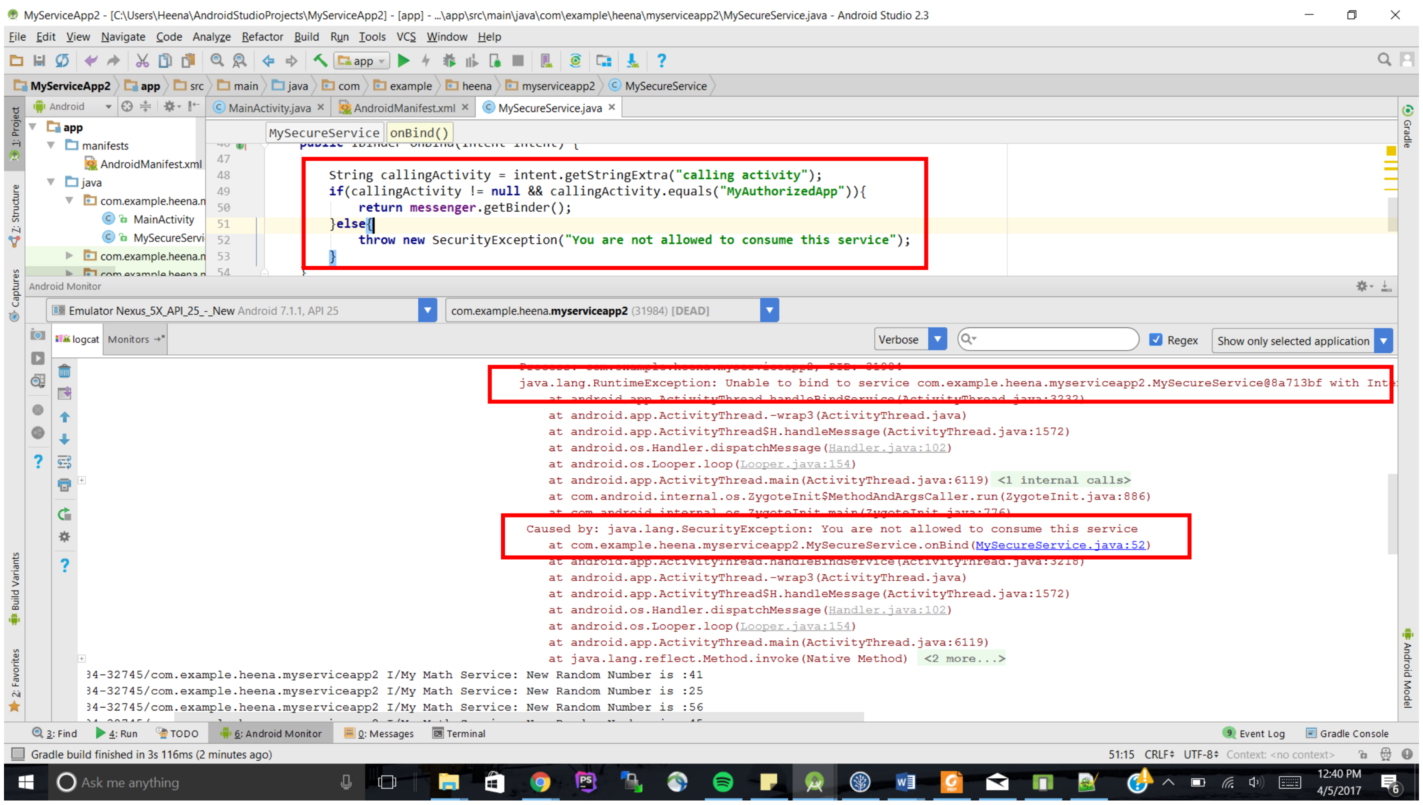
Android Emulator - Nexus_5X_API_25_-_New:5554

Emulator Nexus_5X_API_25_-_New Android 7.1.1, API 25 | com.example.heena.myclientapp2 (32164)

MyServiceApp2 has stopped

Open app again

Gradle build finished in 1s 467ms (a minute ago)



MyServiceApp2 - [C:\Users\Heena\AndroidStudioProjects\MyServiceApp2] - [app] - ...\app\src\main\java\com\example\heena\myserviceapp2\MySecureService.java - Android Studio 2.3

```java
    String callingActivity = intent.getStringExtra("calling activity");
    if(callingActivity != null && callingActivity.equals("MyAuthorizedApp")){
        return messenger.getBinder();
    }else{
        throw new SecurityException("You are not allowed to consume this service");
    }
```

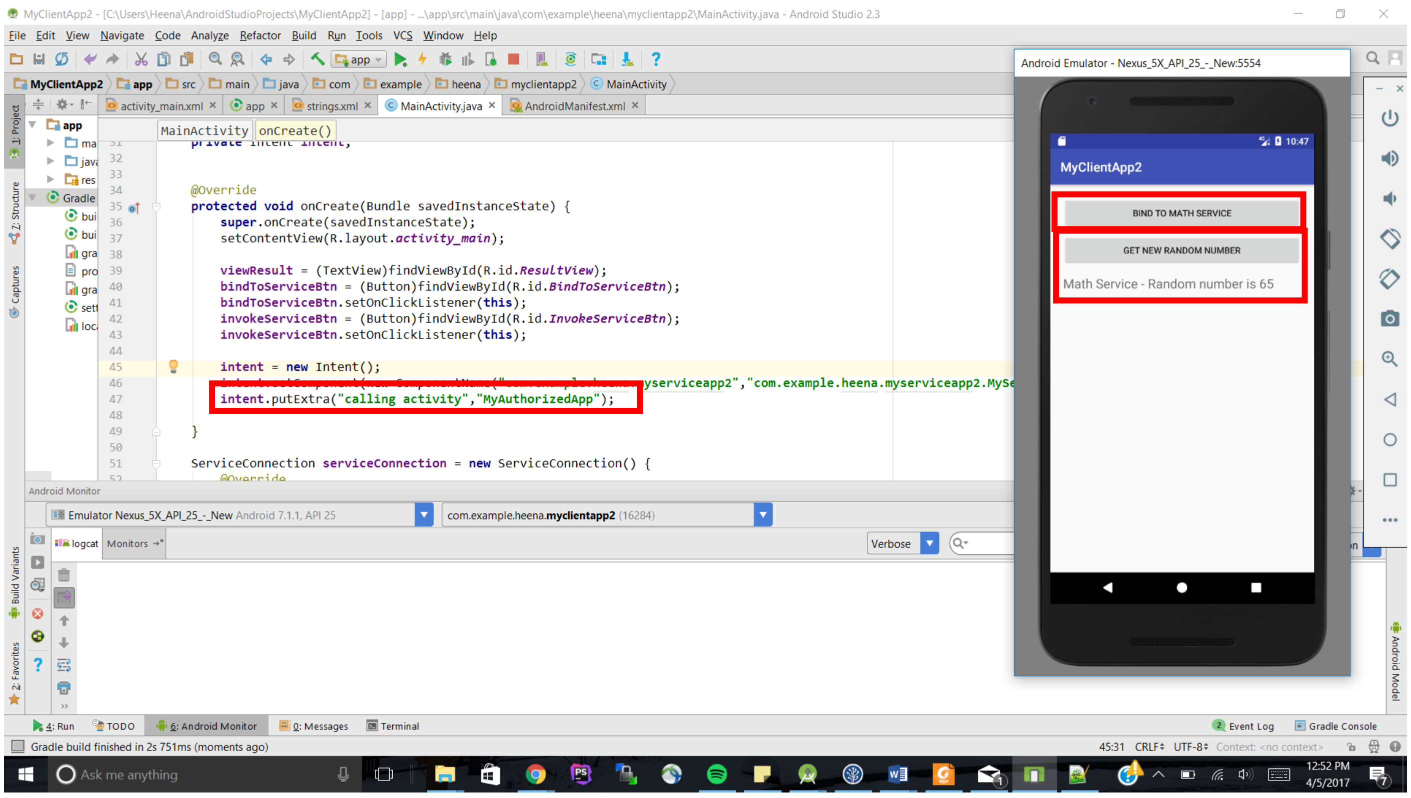
Emulator Nexus_5X_API_25_-_New Android 7.1.1, API 25 | com.example.heena.myserviceapp2 (31984) [DEAD]

Verbose | Regex ✓ | Show only selected application

```
Process: com.example.heena.myserviceapp2, PID: 31984
java.lang.RuntimeException: Unable to bind to service com.example.heena.myserviceapp2.MySecureService@8a713bf with Inte
    at android.app.ActivityThread.handleBindService(ActivityThread.java:3232)
    at android.app.ActivityThread.-wrap3(ActivityThread.java)
    at android.app.ActivityThread$H.handleMessage(ActivityThread.java:1572)
    at android.os.Handler.dispatchMessage(Handler.java:102)
    at android.os.Looper.loop(Looper.java:154)
    at android.app.ActivityThread.main(ActivityThread.java:6119) <1 internal calls>
    at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:886)
    at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:776)
 Caused by: java.lang.SecurityException: You are not allowed to consume this service
    at com.example.heena.myserviceapp2.MySecureService.onBind(MySecureService.java:52)
    at android.app.ActivityThread.handleBindService(ActivityThread.java:3218)
    at android.app.ActivityThread.-wrap3(ActivityThread.java)
    at android.app.ActivityThread$H.handleMessage(ActivityThread.java:1572)
    at android.os.Handler.dispatchMessage(Handler.java:102)
    at android.os.Looper.loop(Looper.java:154)
    at android.app.ActivityThread.main(ActivityThread.java:6119)
    at java.lang.reflect.Method.invoke(Native Method) <2 more...>
34-32745/com.example.heena.myserviceapp2 I/My Math Service: New Random Number is :41
34-32745/com.example.heena.myserviceapp2 I/My Math Service: New Random Number is :25
34-32745/com.example.heena.myserviceapp2 I/My Math Service: New Random Number is :56
```

Gradle build finished in 3s 116ms (2 minutes ago)

On adding the parameter to the intent in the Client app we should be able to bind to the service and start the service to receive data from the Math Service



```
    private Intent intent;


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

        viewResult = (TextView)findViewById(R.id.ResultView);
        bindToServiceBtn = (Button)findViewById(R.id.BindToServiceBtn);
        bindToServiceBtn.setOnClickListener(this);
        invokeServiceBtn = (Button)findViewById(R.id.InvokeServiceBtn);
        invokeServiceBtn.setOnClickListener(this);

        intent = new Intent();
        intent.setComponent(new ComponentName("com.example.heena.myserviceapp2","com.example.heena.myserviceapp2.MySe
        intent.putExtra("calling activity","MyAuthorizedApp");

    }

    ServiceConnection serviceConnection = new ServiceConnection() {
        @Override
```

MyClientApp2

Math Service - Random number is 65

Gradle build finished in 2s 751ms (moments ago)