**Broadcast**

Description:

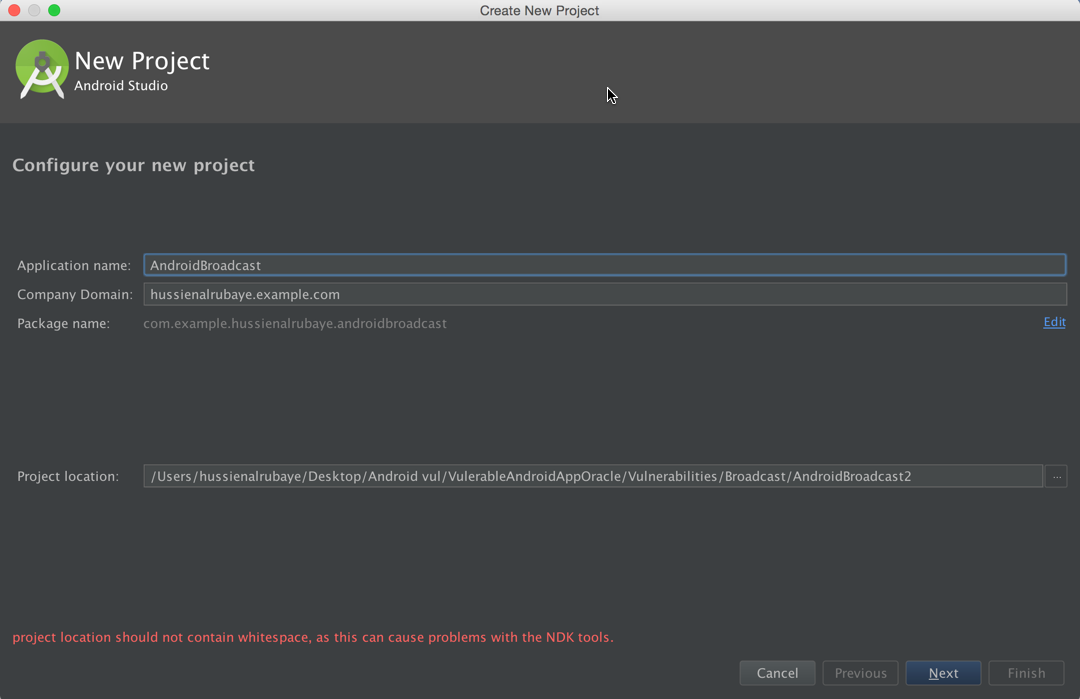
Android has events named “Broadcast events” that events happens when receiving messages or receiving calls in the phone. Android send broadcast in the system saying “hey, this events is happening now and here is the events data” such as messages content or caller phone number. Any apps in the system could listen to this events and read this data, that is mean that we could develop apps blocks phone calls from specific people by listen to the broadcast phone calls, and when some one call, we could end his call.

The interesting thing here is that Android allows developers to build apps that receive and send broadcast events to the system for specific purpose. For example, most of apps has **services** that run in background getting data from the server or do alerts or whatever, so we cannot call activities from background, we have to send broadcast from **services to the broadcast class** then run the activity from broadcast class.

Here the security point we will investigate on. When we send broadcast data we have to make sure that who will receive this data and how to protect it. Especially when we send sensitive data like user name, password, user location and phone number. We have to know that everyone could listen to our broadcast sending data and read it. We will build software to send broadcast username from services then we read in in app broadcast to display in message, then we will build another hacker apps that will read this data, when we broadcast it, then we protect the sending data from hacker.

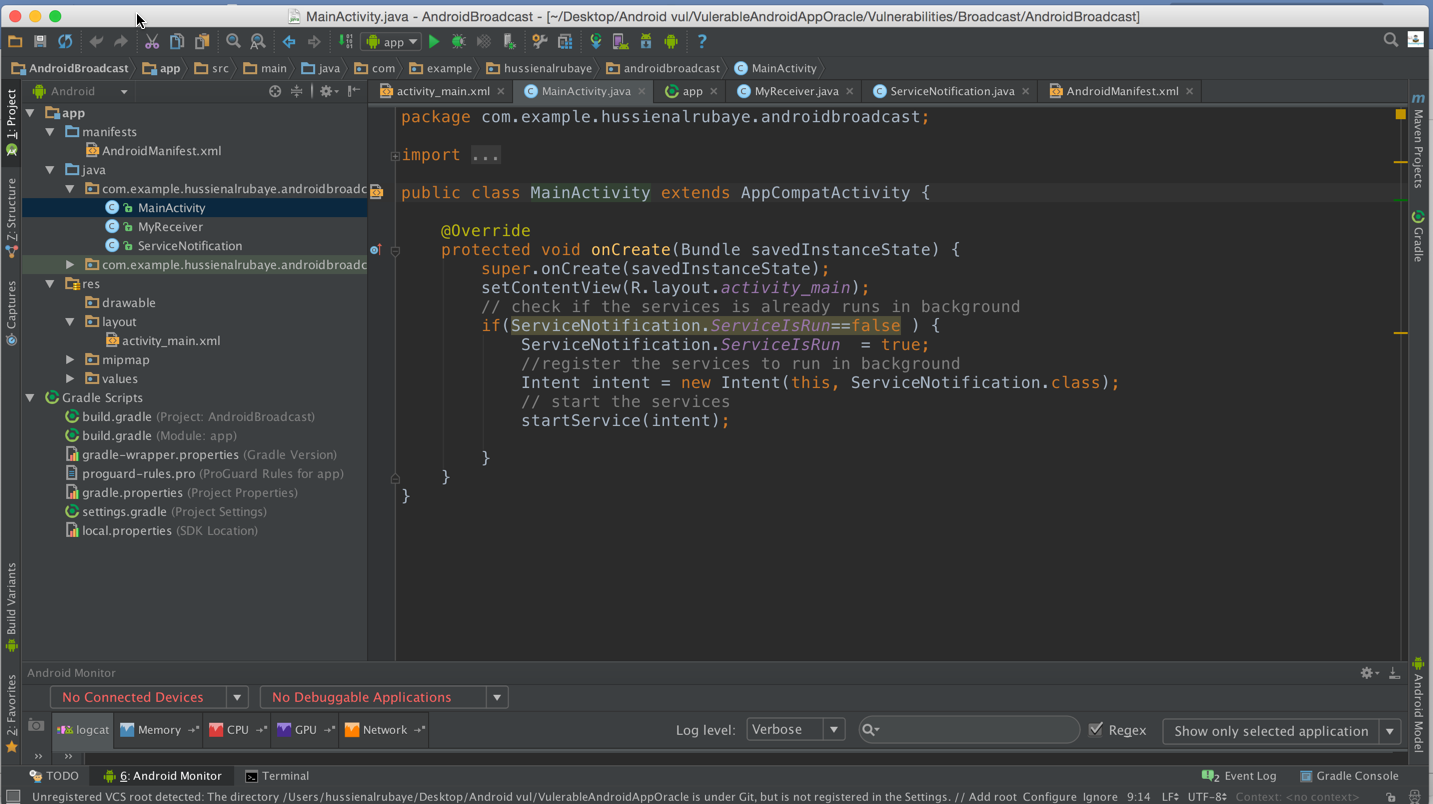
**Steps to build the App:** this app will send user name every 50000ms from the services to system broadcast. And there is Broadcast class in the app which listen to this sending and read the data, then the broadcast class displays the data in **Toast** message.

1. Open new project with name “AndroidBroadcast”, save the package name will will need next



1. Build three classes

* **MainActivity** – start the services
* **MyReceiver**- listen for broadcast events
* **ServiceNotification**—service that send user name to broadcast every 50000ms



1. Build broadcast class named “MyReceiver.java” to receive the action and display the data

|  |
| --- |
| Java |
| import android.content.BroadcastReceiver; import android.content.Context; import android.content.Intent; import android.os.Bundle; import android.widget.Toast;  */\*\*  \* Created by hussienalrubaye on 3/6/16.  \*/* public class MyReceiver extends BroadcastReceiver {   @Override  public void onReceive(Context context, Intent intent) {  // get the bundles in the message  final Bundle bundle = intent.getExtras();  // check the action equal to the action we fire in broadcast,  if ( intent.getAction().equalsIgnoreCase("com.example.Broadcast"))  //read the data from the intent  Toast.*makeText*(context,bundle.getString("username"),Toast.*LENGTH\_LONG*).show();  } } |

1. Build service class named “ServiceNotification.java” to send user name “alxs1aa” as broadcast every 50000ms

|  |
| --- |
| Java |
| import android.app.IntentService; import android.content.Intent;  import org.json.JSONArray; import org.json.JSONObject;  import java.io.BufferedInputStream; import java.io.InputStream; import java.net.HttpURLConnection; import java.net.URL;  */\*\*  \* Created by hussienalrubaye on 3/6/16.  \* this services send broadcast messages every 50000ms  \*/* public class ServiceNotification extends IntentService {  public static boolean *ServiceIsRun*=false;   public ServiceNotification() {  super("MyWebRequestService");  }  protected void onHandleIntent(Intent workIntent) {   // continue sending the messages  while ( *ServiceIsRun*) {  // creat new intent  Intent intent = new Intent();  //set the action that will receive our broadcast  intent.setAction("com.example.Broadcast");  // add data to the bundle  intent.putExtra("username", "alxs1aa");  // send the data to broadcast  sendBroadcast(intent);  //delay for 50000ms  try{  Thread.*sleep*(50000);  }catch (Exception ex){}    }  }   } |

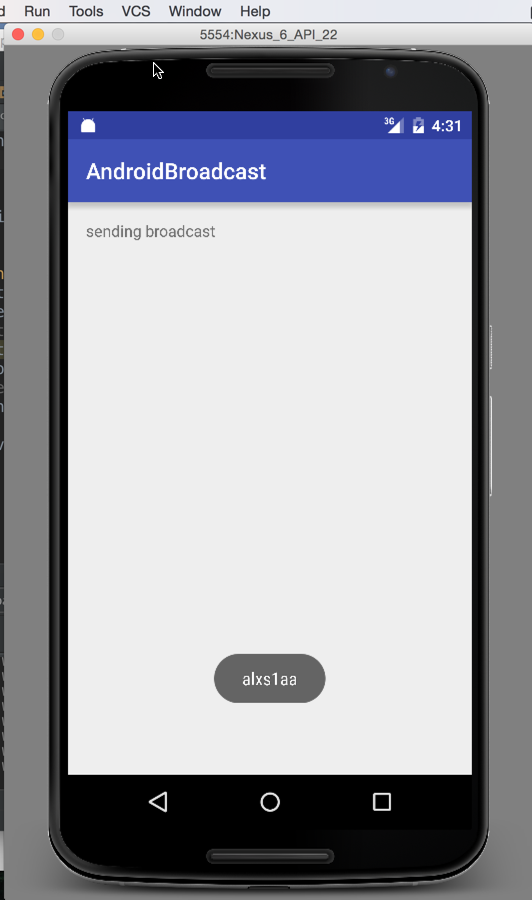
1. Write this code in “MainActivty.java” Start the service

|  |
| --- |
| Java |
| import android.content.Intent; import android.support.v7.app.AppCompatActivity; import android.os.Bundle; import android.view.View; import android.widget.Button;  public class MainActivity extends AppCompatActivity {   @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_main*);  // check if the services is already runs in background  if(ServiceNotification.*ServiceIsRun*==false ) {  ServiceNotification.*ServiceIsRun* = true;  //register the services to run in background  Intent intent = new Intent(this, ServiceNotification.class);  // start the services  startService(intent);   }  } } |

1. Register the service and broadcast in the manifest.xml to be like this

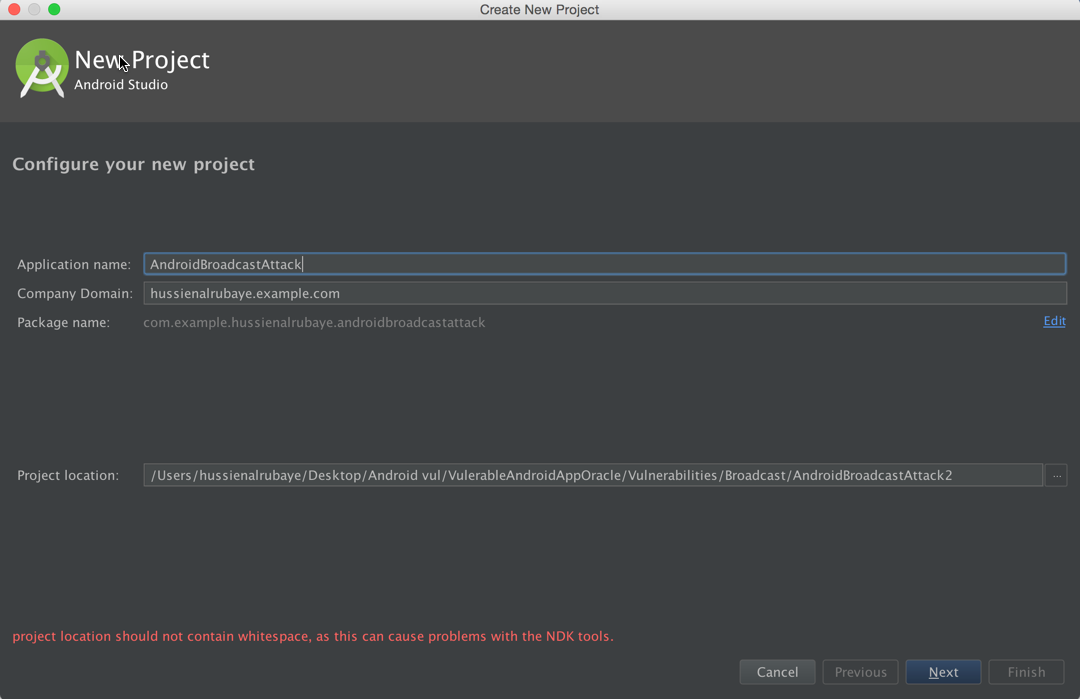
|  |
| --- |
| Java |
| <?xml version="1.0" encoding="utf-8"?> <manifest xmlns:android="http://schemas.android.com/apk/res/android"  package="com.example.hussienalrubaye.androidbroadcast">   <application  android:allowBackup="true"  android:icon="@mipmap/ic\_launcher"  android:label="@string/app\_name"  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>  *<!-- register the broadcast to listen to action names com.example.Broadcast-->* <receiver android:name=".MyReceiver" android:priority="2147483647" >  <intent-filter>  <action android:name="com.example.Broadcast" >  </action>  </intent-filter>  </receiver>  *<!-- register the service-->* <service  android:name=".ServiceNotification"  android:exported="false" >  </service>  </application>  </manifest> |

Run the app you will see it display message every 5000ms that has “alxs1aa”



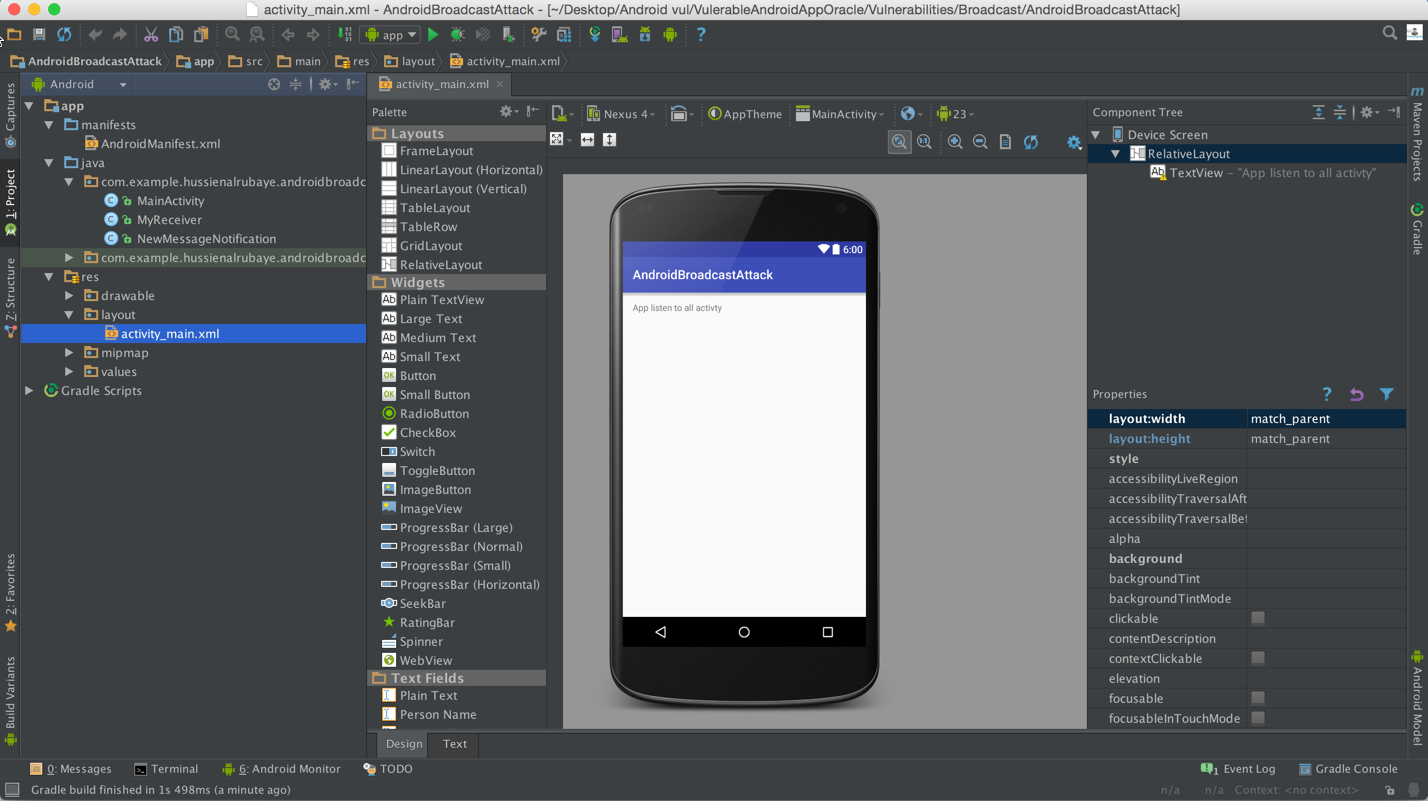
**Steps to build hacker app:** it easy to any hacker to unzip your APK and read the app mainest.xml then he will build apps to listen too same broadcast action that you listen too in his app and read your data, we will build apps for doing this job

1. Create new app named “AndroidBroadcastAttack”



1. Build three classes

* **MainActivity** – start the services
* **MyReceiver**- listen for broadcast events
* **NewMessageNotification**—add from new->UI component->**Notification**



1. Create class named “MyReceiver” Listen to broadcast messages and display notification

|  |
| --- |
| Java |
| import android.content.BroadcastReceiver; import android.content.Context; import android.content.Intent; import android.os.Bundle; import android.widget.Toast;  */\*\*  \* Created by hussienalrubaye on 3/6/16.  \*/* public class MyReceiver extends BroadcastReceiver {   @Override  public void onReceive(Context context, Intent intent) {  String DataBundel="";  // get app the data sent on bundle  Bundle bundle= intent.getExtras();  //lopp through all keys in the bundle  for (String key : bundle.keySet()) {  // get object by key( we define object became it may be text or image or whatever  Object value = bundle.get(key);  //get all keys  DataBundel+= String.*format*("%s %s (%s)", key, value.toString(), value.getClass().getName());  }  //display notify message to the user  NewMessageNotification NotifyMe=new NewMessageNotification();  NotifyMe.*notify*( context, DataBundel, 123);   } } |

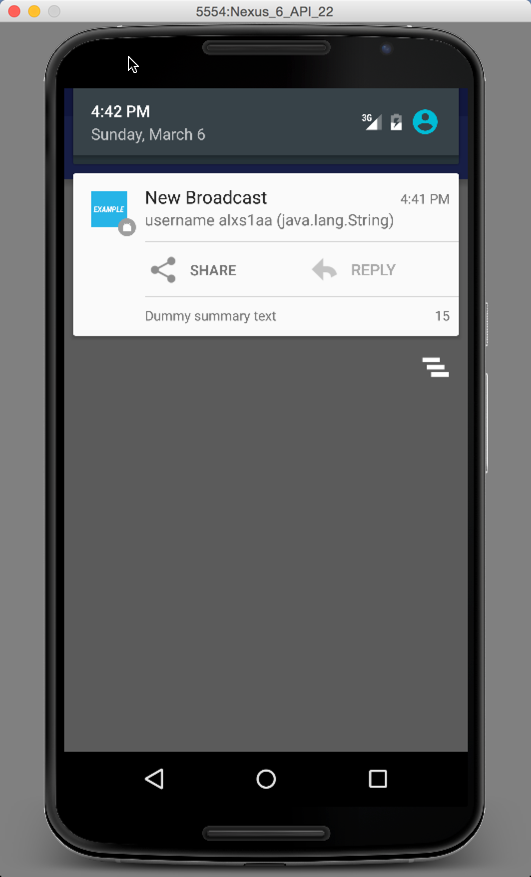
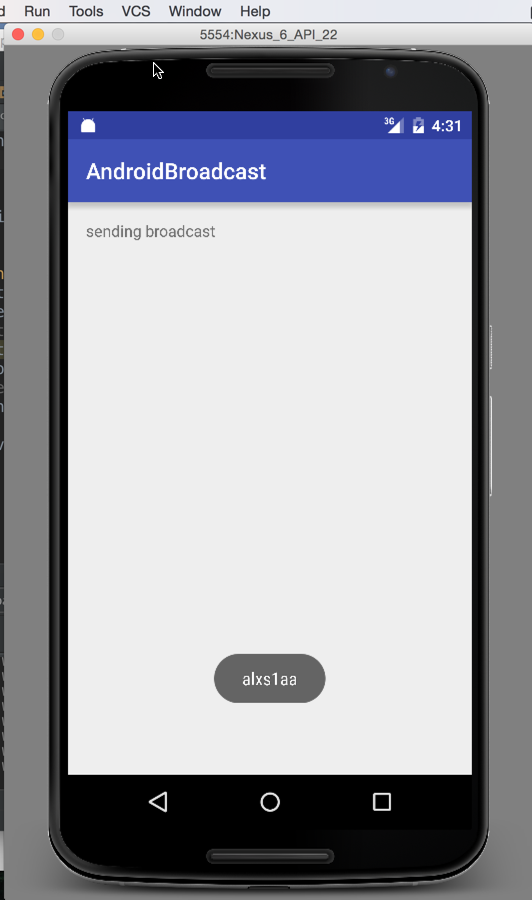
1. Create class named “NewMessageNotification” to display notification

|  |
| --- |
| Java |
| import android.annotation.TargetApi; import android.app.Notification; import android.app.NotificationManager; import android.app.PendingIntent; import android.content.Context; import android.content.Intent; import android.content.res.Resources; import android.graphics.Bitmap; import android.graphics.BitmapFactory; import android.net.Uri; import android.os.Build; import android.support.v4.app.NotificationCompat;  */\*\*  \* Helper class for showing and canceling new message  \* notifications.  \* <p>  \* This class makes heavy use of the {****@link*** *NotificationCompat.Builder} helper  \* class to create notifications in a backward-compatible way.  \*/* public class NewMessageNotification {  */\*\*  \* The unique identifier for this type of notification.  \*/* private static final String *NOTIFICATION\_TAG* = "NewMessage";   */\*\*  \* Shows the notification, or updates a previously shown notification of  \* this type, with the given parameters.  \* <p>  \* TODO: Customize this method's arguments to present relevant content in  \* the notification.  \* <p>  \* TODO: Customize the contents of this method to tweak the behavior and  \* presentation of new message notifications. Make  \* sure to follow the  \* <a href="https://developer.android.com/design/patterns/notifications.html">  \* Notification design guidelines</a> when doing so.  \*  \** ***@see*** *#cancel(Context)  \*/* public static void notify(final Context context,  final String exampleString, final int number) {  final Resources res = context.getResources();   // This image is used as the notification's large icon (thumbnail).  // *TODO: Remove this if your notification has no relevant thumbnail.* final Bitmap picture = BitmapFactory.*decodeResource*(res, R.drawable.*example\_picture*);    final String ticker = exampleString;  final String title = "New Broadcast";  final String text =exampleString;   final NotificationCompat.Builder builder = new NotificationCompat.Builder(context)   // Set appropriate defaults for the notification light, sound,  // and vibration.  .setDefaults(Notification.*DEFAULT\_ALL*)   // Set required fields, including the small icon, the  // notification title, and text.  .setSmallIcon(R.drawable.*ic\_stat\_new\_message*)  .setContentTitle(title)  .setContentText(text)   // All fields below this line are optional.   // Use a default priority (recognized on devices running Android  // 4.1 or later)  .setPriority(NotificationCompat.*PRIORITY\_DEFAULT*)   // Provide a large icon, shown with the notification in the  // notification drawer on devices running Android 3.0 or later.  .setLargeIcon(picture)   // Set ticker text (preview) information for this notification.  .setTicker(ticker)   // Show a number. This is useful when stacking notifications of  // a single type.  .setNumber(number)   // If this notification relates to a past or upcoming event, you  // should set the relevant time information using the setWhen  // method below. If this call is omitted, the notification's  // timestamp will by set to the time at which it was shown.  // *TODO: Call setWhen if this notification relates to a past or* // upcoming event. The sole argument to this method should be  // the notification timestamp in milliseconds.  //.setWhen(...)   // Set the pending intent to be initiated when the user touches  // the notification.  .setContentIntent(  PendingIntent.*getActivity*(  context,  0,  new Intent(Intent.*ACTION\_VIEW*, Uri.*parse*("http://www.google.com")),  PendingIntent.*FLAG\_UPDATE\_CURRENT*))   // Show expanded text content on devices running Android 4.1 or  // later.  .setStyle(new NotificationCompat.BigTextStyle()  .bigText(text)  .setBigContentTitle(title)  .setSummaryText("Dummy summary text"))   // Example additional actions for this notification. These will  // only show on devices running Android 4.1 or later, so you  // should ensure that the activity in this notification's  // content intent provides access to the same actions in  // another way.  .addAction(  R.drawable.*ic\_action\_stat\_share*,  res.getString(R.string.*action\_share*),  PendingIntent.*getActivity*(  context,  0,  Intent.*createChooser*(new Intent(Intent.*ACTION\_SEND*)  .setType("text/plain")  .putExtra(Intent.*EXTRA\_TEXT*, "Dummy text"), "Dummy title"),  PendingIntent.*FLAG\_UPDATE\_CURRENT*))  .addAction(  R.drawable.*ic\_action\_stat\_reply*,  res.getString(R.string.*action\_reply*),  null)   // Automatically dismiss the notification when it is touched.  .setAutoCancel(true);   *notify*(context, builder.build());  }   @TargetApi(Build.VERSION\_CODES.*ECLAIR*)  private static void notify(final Context context, final Notification notification) {  final NotificationManager nm = (NotificationManager) context  .getSystemService(Context.*NOTIFICATION\_SERVICE*);  if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*ECLAIR*) {  nm.notify(*NOTIFICATION\_TAG*, 0, notification);  } else {  nm.notify(*NOTIFICATION\_TAG*.hashCode(), notification);  }  }   */\*\*  \* Cancels any notifications of this type previously shown using  \* {****@link*** *#notify(Context, String, int)}.  \*/* @TargetApi(Build.VERSION\_CODES.*ECLAIR*)  public static void cancel(final Context context) {  final NotificationManager nm = (NotificationManager) context  .getSystemService(Context.*NOTIFICATION\_SERVICE*);  if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*ECLAIR*) {  nm.cancel(*NOTIFICATION\_TAG*, 0);  } else {  nm.cancel(*NOTIFICATION\_TAG*.hashCode());  }  } } |

1. Register Broadcast in the MainFast.xml

|  |
| --- |
| Java |
| <?xml version="1.0" encoding="utf-8"?> <manifest xmlns:android="http://schemas.android.com/apk/res/android"  package="com.example.hussienalrubaye.androidbroadcastattack">   <uses-permission android:name="android.permission.VIBRATE" />   <application  android:allowBackup="true"  android:icon="@mipmap/ic\_launcher"  android:label="@string/app\_name"  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>  *<!-- Register broadcast to listen to broadcast action names com.example.Broadcast-->* <receiver  android:name=".MyReceiver"  android:priority="2147483647">  <intent-filter>  <action android:name="com.example.Broadcast"></action>  </intent-filter>  </receiver>  </application>  </manifest> |

Whenever broadcast send hacker app will read it and display it in notification



**Fix This Problem**

To fix this problem use encryption and decryption theory. That is mean that in sending broadcast we encrypt the data over the bundle then we decrypt it when we receive it

**To do that we will update our app to have encryption doing this steps**

1. **Add new class named “security.java”**

|  |
| --- |
| 1. Java |
| */\*\*  \* Created by hussienalrubaye on 3/6/16.  \*/* public class Security {   // cipher encryption add shift for key  public static 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;  } } |

2-update broadcast class named “MyReceiver.java”

|  |
| --- |
| Java |
| import android.content.BroadcastReceiver; import android.content.Context; import android.content.Intent; import android.os.Bundle; import android.widget.Toast;  */\*\*  \* Created by hussienalrubaye on 3/6/16.  \*/* public class MyReceiver extends BroadcastReceiver {   @Override  public void onReceive(Context context, Intent intent) {  // get the bundles in the message  final Bundle bundle = intent.getExtras();  // check the action equal to the action we fire in broadcast,  if ( intent.getAction().equalsIgnoreCase("com.example.Broadcast"))  //read the data from the intent  Toast.*makeText*(context, Security.*cipher*(bundle.getString("username"),-10), Toast.*LENGTH\_LONG*).show();  } } |

1. update service class named “ServiceNotification.java”

|  |
| --- |
| Java |
| import android.app.IntentService; import android.content.Intent;  import org.json.JSONArray; import org.json.JSONObject;  import java.io.BufferedInputStream; import java.io.InputStream; import java.net.HttpURLConnection; import java.net.URL;  */\*\*  \* Created by hussienalrubaye on 3/6/16.  \* this services send broadcast messages every 50000ms  \*/* public class ServiceNotification extends IntentService {  public static boolean *ServiceIsRun*=false;   public ServiceNotification() {  super("MyWebRequestService");  }  protected void onHandleIntent(Intent workIntent) {   // continue sending the messages  while ( *ServiceIsRun*) {  // creat new intent  Intent intent = new Intent();  //set the action that will receive our broadcast  intent.setAction("com.example.Broadcast");  // add data to the bundle  intent.putExtra("username",Security.*cipher*( "alxs1aa",10));  // send the data to broadcast  sendBroadcast(intent);  //delay for 50000ms  try{  Thread.*sleep*(50000);  }catch (Exception ex){}    }  }   } |

After encryption we see event the hacker access to the data but he cannot read it

