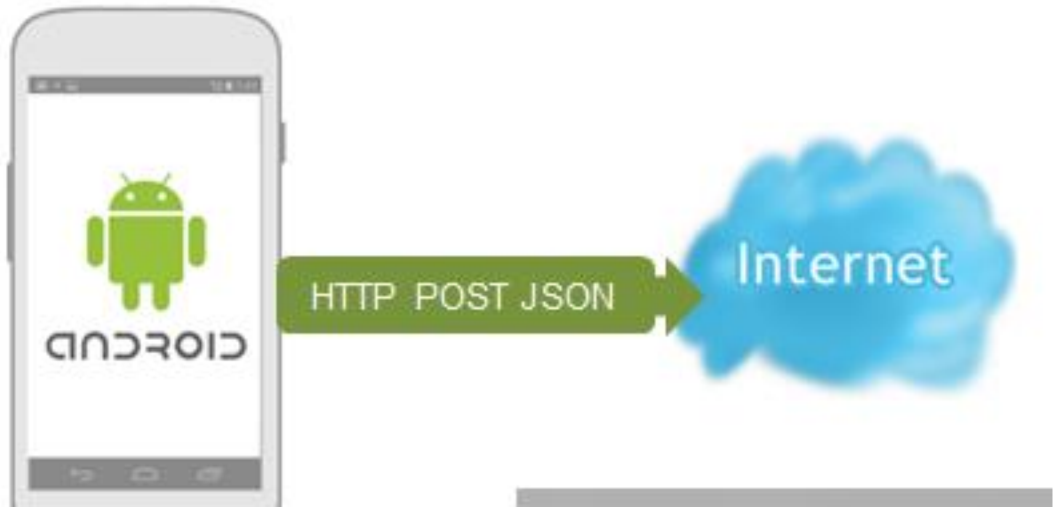
Secure HTTP Requests

Background:

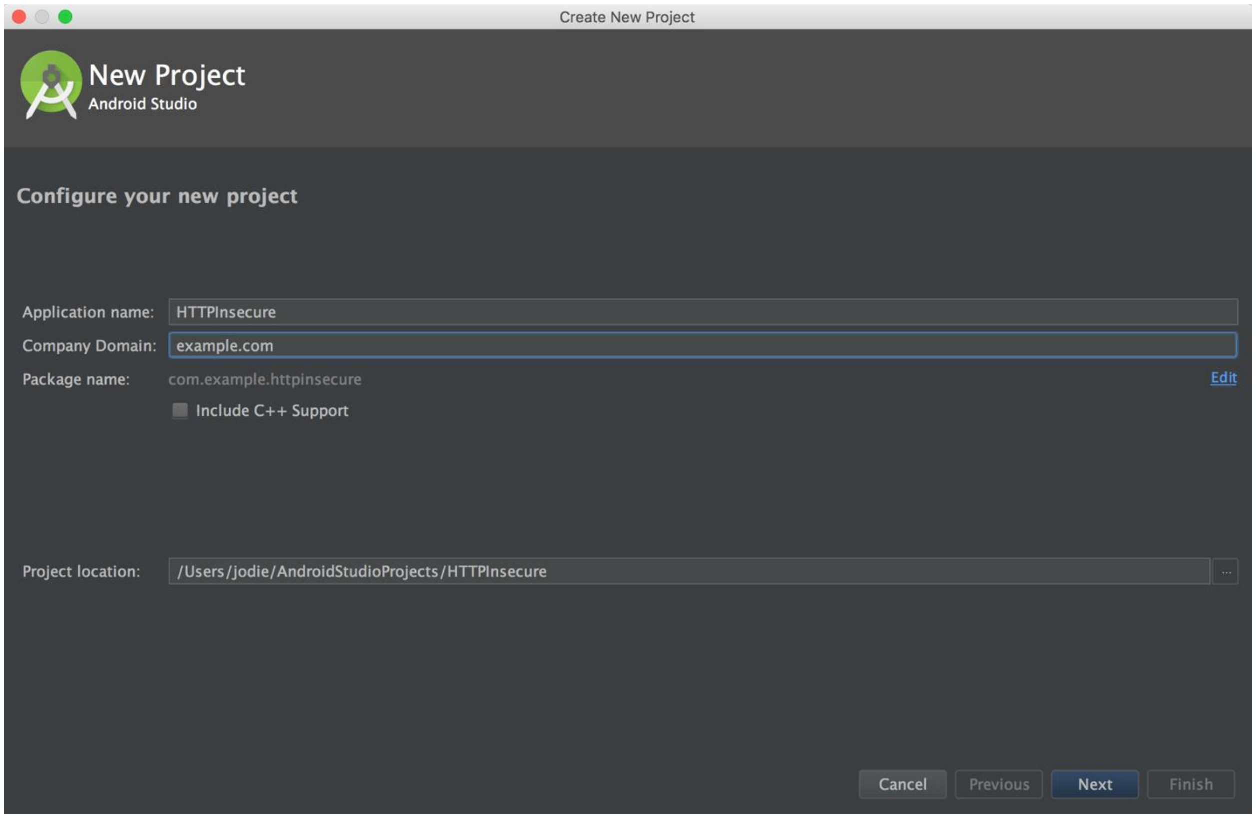
In Android, we use HTTP to exchange data between the client and the server. Sometimes, we send sensitive data, like username and password or user location from client to server over HTTP. Keep in mind that this request will move over the network. When this data is in transit, it is very easy for a hacker to intercept. Many people could interrupt and read this request data, so we must protect it.

We will demonstrate an example of how to send sensitive data between client to server over HTTP. We will then explain how hackers could read this data, along with suggestions of how to protect against this.



Steps to build the app:

1. In Android Studio, create a new project, named "HTTPInsecure". Take note of the package name, as we will be needing this later in the tutorial. Click next.



Application name: HTTPInsecure

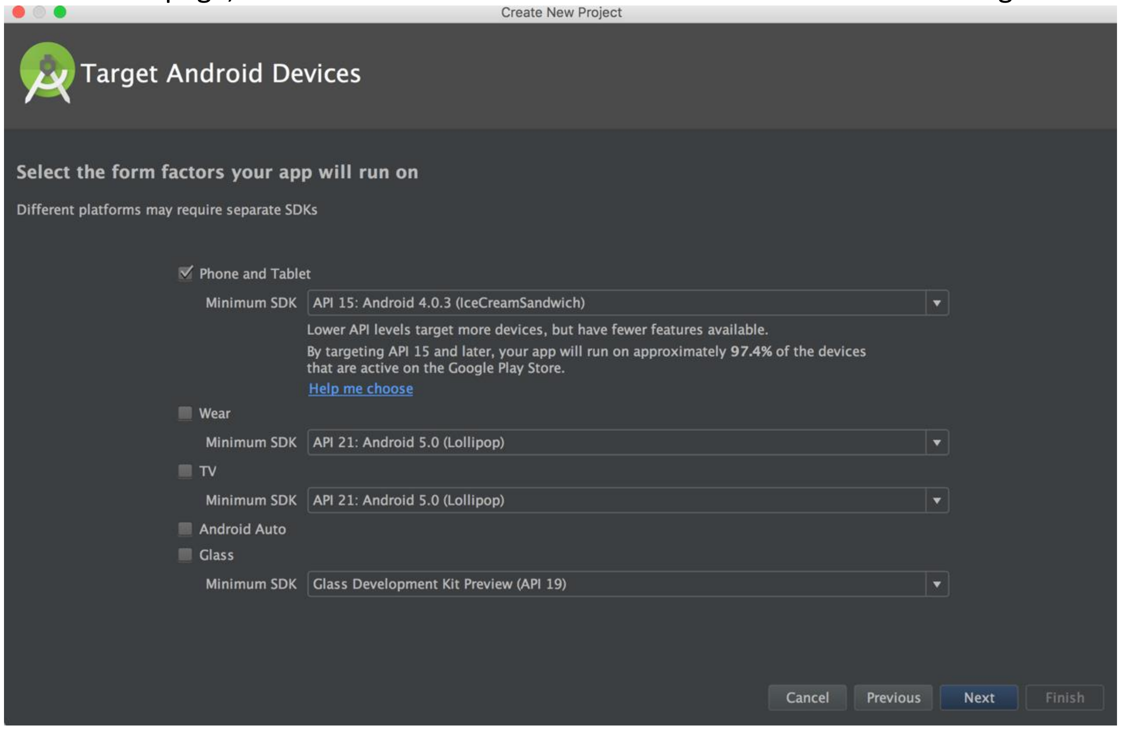
Company Domain: example.com

Package name: com.example.httpinsecure

Include C++ Support

Project location: /Users/jodie/AndroidStudioProjects/HTTPInsecure

On the next page, click next. The default API level will suffice for our needs right now.



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

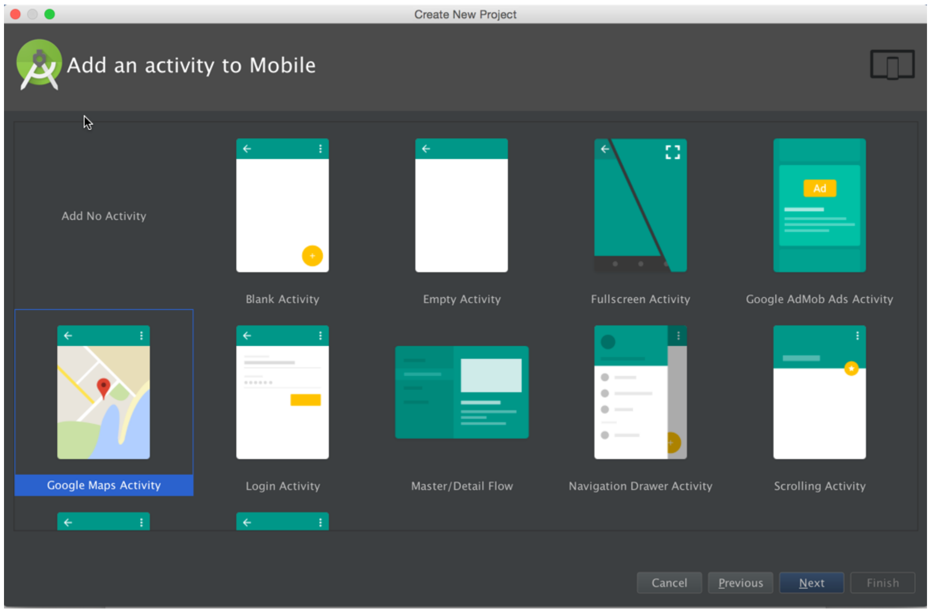
Minimum SDK: API 21: Android 5.0 (Lollipop)

☐ Android Auto

☐ Glass

Minimum SDK: Glass Development Kit Preview (API 19)

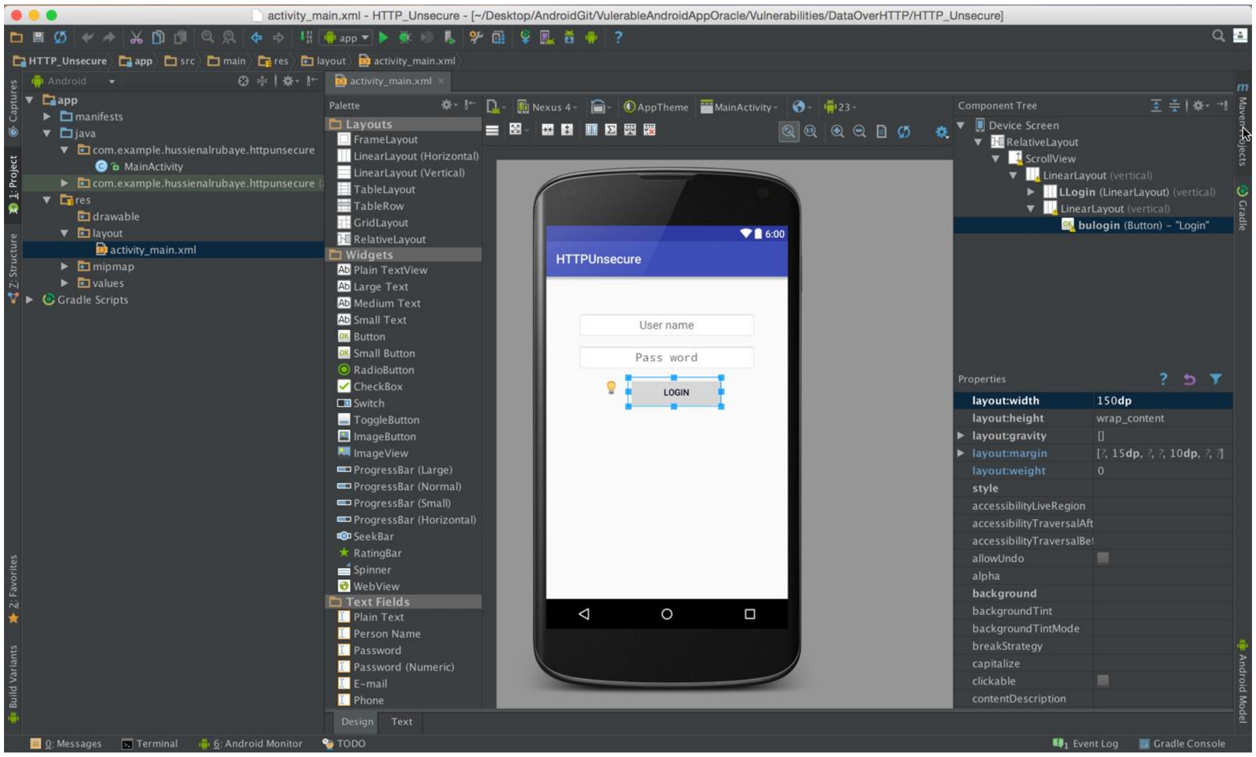
2. Select project type “Empty Activity”.



Create New Project

Add an activity to Mobile

3. Our project will look like this when we are finished with the layout.



```
layout:width                   150dp
layout:height                  wrap_content
layout:gravity                 []
layout:margin                  [?, 15dp, ?, ?, 10dp, ?, ?]
layout:weight                  0
style
accessibilityLiveRegion
accessibilityTraversalAft
accessibilityTraversalBe
allowUndo
alpha
background
backgroundTint
backgroundTintMode
breakStrategy
capitalize
clickable
contentDescription
```

To do this, we will update **activity_main.xml** with the code below. The **activity_main.xml** file can be found under "res/layout" in the folders in the panel to the left.
Click the "Text" tab, found in the middle near the bottom of the window to see the text view of the XML file. By default, it shows you the Design view first, which is what's currently visible in the picture above.

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:paddingBottom="@dimen/activity_vertical_margin"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  tools:context="com.example.httpinsecure.MainActivity">

  <ScrollView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1">
```

```
<LinearLayout
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingTop="30dp"
    android:paddingLeft="15dp"
    android:paddingRight="15dp">

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/LLogin"
        android:layout_marginTop="7dp"
        android:paddingLeft="20dp"
        android:paddingTop="4dp"
        android:paddingRight="20dp">

        <LinearLayout
            android:orientation="vertical"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:touchscreenBlocksFocus="false"
            android:layout_marginBottom="2dp">

            <EditText
                android:gravity="center"
                android:maxLength="50"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:inputType="text"
                android:ems="10"
                android:id="@+id/EDTUserName"
                android:layout_weight="1"
                android:textColor="#ff1a102c"
                android:background="@android:drawable/editbox_background"
                android:textSize="18dp"
                android:hint="Username"
                android:paddingBottom="9dp"
                android:paddingTop="9dp"
                android:layout_marginBottom="10dp"/>
        </LinearLayout>



        <LinearLayout
            android:orientation="vertical"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:touchscreenBlocksFocus="false"
```

```
                android:layout_marginBottom="2dp">

                <EditText
                    android:gravity="center"
                    android:maxLength="50"
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
                    android:ems="10"
                    android:id="@+id/EDTpassword"
                    android:layout_weight="1"
                    android:textColor="#ff1a102c"
                    android:background="@android:drawable/editbox_background"
                    android:textSize="18dp"
                    android:hint="Password"
                    android:paddingBottom="9dp"
                    android:paddingTop="9dp"
                    android:layout_marginBottom="10dp"
                    android:inputType="textPassword" />
            </LinearLayout>
        </LinearLayout>

        <LinearLayout
            android:textAlignment="center"
            android:gravity="center"
            android:orientation="vertical"
            android:layout_width="match_parent"
            android:layout_height="match_parent">
            <Button
                android:layout_width="150dp"
                android:layout_height="wrap_content"
                android:text="Login"
                android:id="@+id/bulogin"
                android:drawablePadding="4dp"
                android:layout_marginBottom="10dp"
                android:textColor="#ff06071c"
                android:onClick="buloginckic"
                android:layout_weight="0"
                android:layout_marginLeft="15dp" />
        </LinearLayout>
    </LinearLayout>
  </ScrollView>
</RelativeLayout>
```

4. Next, update the **MainActivity.java** file with the code below. The **MainActivity.java** file can be found under the path “app/java/com.example.httpinsecure”.

```
package com.example.httpinsecure;

import android.support.v7.app.AppCompatActivity;
import android.os.AsyncTask;
import android.os.Bundle;
import android.widget.EditText;
import android.widget.Toast;
import android.view.View;

import java.io.BufferedInputStream;
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void bulogincnckic(View view) {
        //get user name and password
        EditText UserName=(EditText)findViewById(R.id.EDTUserName);
        EditText Password=(EditText)findViewById(R.id.EDTpassword);

        // send user name and password over the http
        String url="http://sellingportal.alruabye.net/UsersWS.asmx/Login?UserName="+
UserName.getText().toString() +"&Password="+ Password.getText().toString();

        // start background task
        new MyAsyncTaskGetNews().execute(url, "news");
    }


    // get news from server
    public class MyAsyncTaskGetNews extends AsyncTask<String, String, String> {
        @Override
        protected void onPreExecute() {
            //before works
        }

        @Override
        protected String doInBackground(String... params) {
            try {
                String NewsData;
                //define the url we have to connect with
                URL url = new URL(params[0]);
                //make connect with url and send request
                HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
                //waiting for 7000ms for response
                urlConnection.setConnectTimeout(7000);//set timeout to 5 seconds
                try {
                    //getting the response data
                    InputStream in = new BufferedInputStream(urlConnection.getInputStream());
                    //convert the stream to string
                    NewsData = ConvertInputToStringNoChange(in);
                    //send to display data
                    publishProgress(NewsData);
                } finally {
                    //end connection
                    urlConnection.disconnect();
                }

            } catch (Exception ex){}
            return null;
        }

        protected void onProgressUpdate(String... progress) {
            try {
                //display response data
                Toast.makeText(getApplicationContext(),progress[0],Toast.LENGTH_LONG).show();
            } catch (Exception ex) {
            }
        }

        protected void onPostExecute(String result2) {

        }
    }

    // this method convert any stream to string
    public static String ConvertInputToStringNoChange(InputStream inputStream) {
        BufferedReader bureader=new BufferedReader( new InputStreamReader(inputStream));
        String line ;
        String linereultcal="";

        try {
            while((line=bureader.readLine())!=null) {

                linereultcal+=line;

            }
            inputStream.close();
        } catch (Exception ex) {}

        return linereultcal;
    }
}
```

5. Add permission to access the internet to **Manifest.xml**, found under “app/manifests/”. The highlighted line is the way we’re going to add the permission.

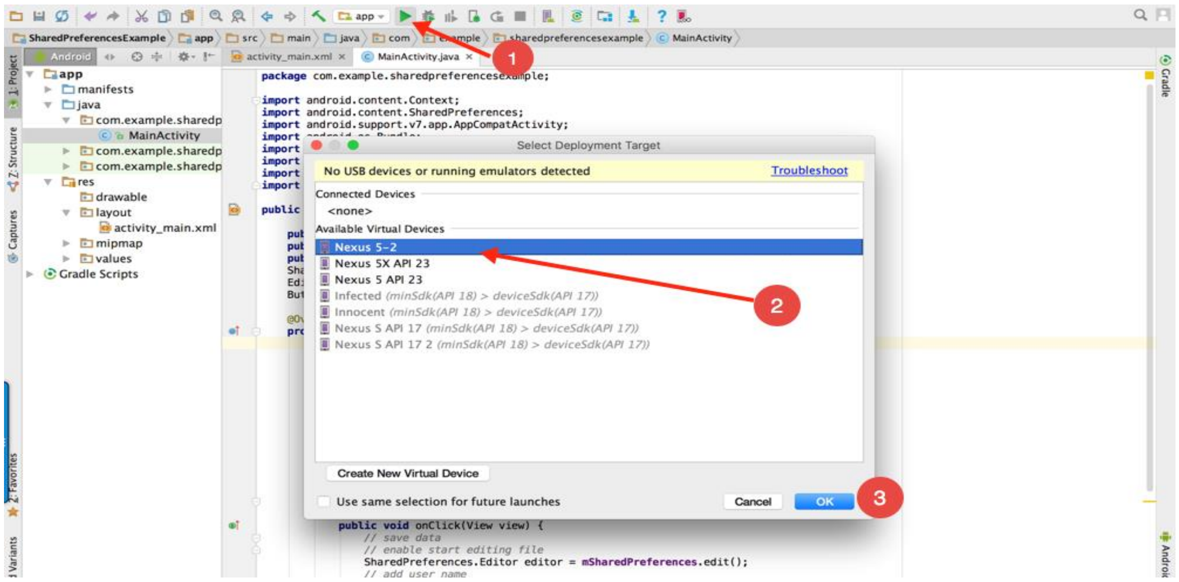
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.httpinsecure">

  <uses-permission android:name="android.permission.INTERNET"></uses-permission>
  <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> <!-- ATTENTION: This was auto-generated to add Google Play services to your project for
  App Indexing.  See https://g.co/AppIndexing/AndroidStudio for more information. -->
    <meta-data
      android:name="com.google.android.gms.version"
      android:value="@integer/google_play_services_version" />
  </application>

</manifest>
```

See the image below to see how to start the virtual device and run the app. You may need to click “Create New Virtual Device” if you have nothing under your list of **Available Virtual Devices.**



```
package com.example.sharedpreferencesexample;

import android.content.Context;
import android.content.SharedPreferences;
import android.support.v7.app.AppCompatActivity;
```

Select Deployment Target

No USB devices or running emulators detected   Troubleshoot

Connected Devices
<none>

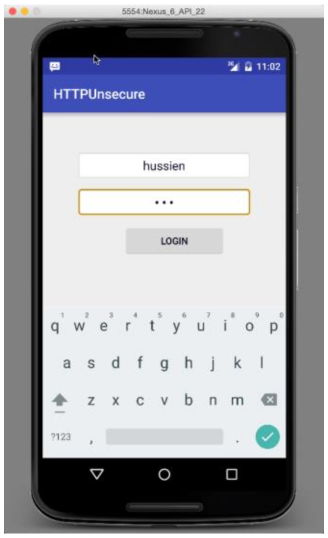
Available Virtual Devices
Nexus 5-2
Nexus 5X API 23
Nexus 5 API 23
Infected (minSdk(API 18) > deviceSdk(API 17))
Innocent (minSdk(API 18) > deviceSdk(API 17))
Nexus S API 17 (minSdk(API 18) > deviceSdk(API 17))
Nexus S API 17 2 (minSdk(API 18) > deviceSdk(API 17))

Create New Virtual Device

Use same selection for future launches   Cancel   OK

```
public void onClick(View view) {
    // save data
    // enable start editing file
    SharedPreferences.Editor editor = mSharedPreferences.edit();
    // add user name
```

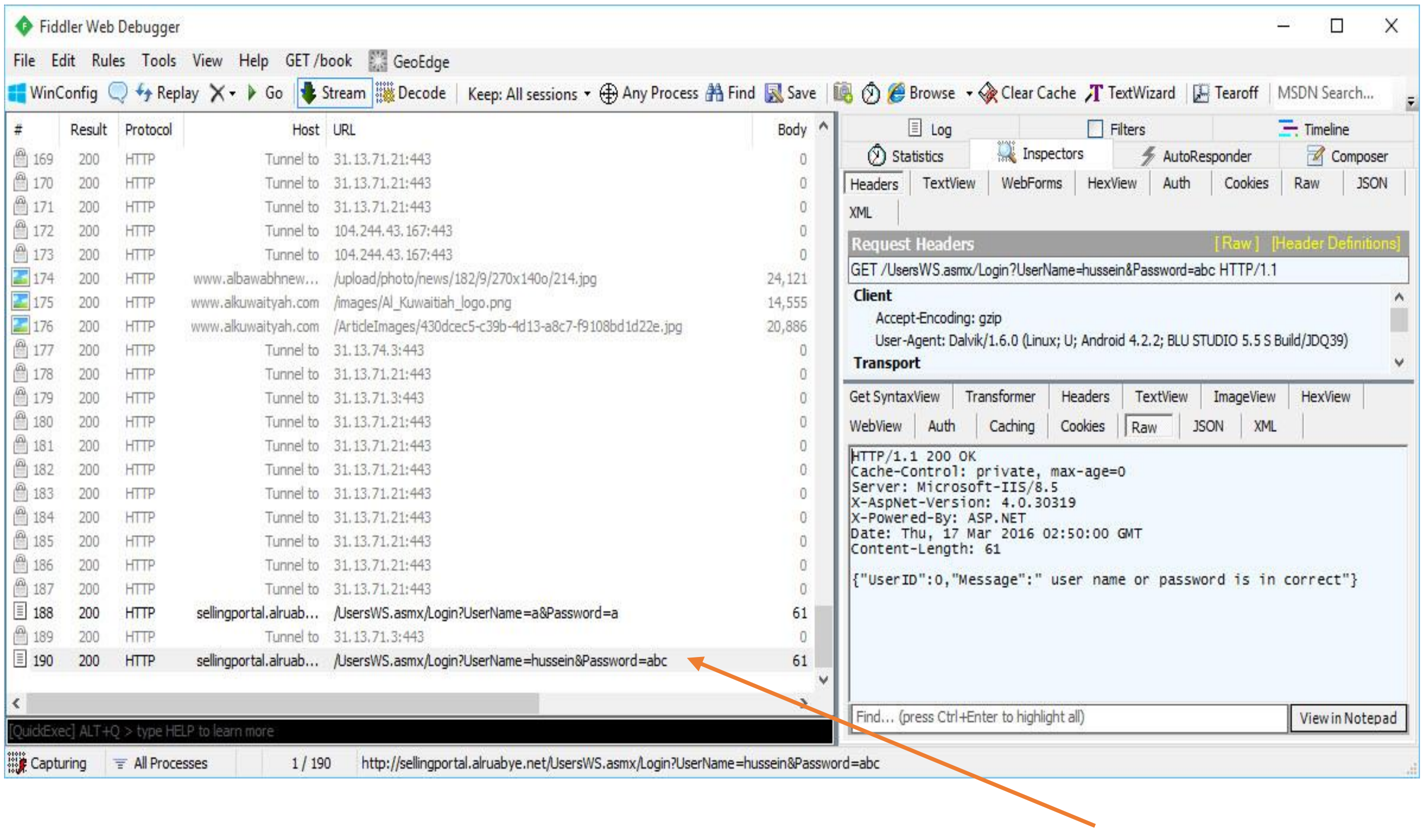
When the app comes up (which may take a while), enter username “Hussein” and password “abc”.



How the data is vulnerable:

When this data is in transit, it is very easy for a hacker to intercept and read the data. We will use Fiddler app to monitor traffic. Click link to install it. Follow the instructions given by the company that makes Fiddler in order to use it.

When we run it, we see that the username and password is available.


Fiddler Web Debugger
File Edit Rules Tools View Help GET /book GeoEdge
WinConfig Replay Go Stream Decode Keep: All sessions Any Process Find Save Browse Clear Cache TextWizard Tearoff MSDN Search...
Log Filters Timeline
Statistics Inspectors AutoResponder Composer
Headers TextView WebForms HexView Auth Cookies Raw JSON XML
Get SyntaxView Transformer Headers TextView ImageView HexView
WebView Auth Caching Cookies Raw JSON XML
Find... (press Ctrl+Enter to highlight all) View in Notepad
[QuickExec] ALT+Q > type HELP to learn more
Capturing All Processes 1 / 190

#	Result	Protocol	Host	URL	Body
169	200	HTTP	Tunnel to	31.13.71.21:443	0
170	200	HTTP	Tunnel to	31.13.71.21:443	0
171	200	HTTP	Tunnel to	31.13.71.21:443	0
172	200	HTTP	Tunnel to	104.244.43.167:443	0
173	200	HTTP	Tunnel to	104.244.43.167:443	0
174	200	HTTP	www.albawabhnew...	/upload/photo/news/182/9/270x140o/214.jpg	24,121
175	200	HTTP	www.alkuwaityah.com	/images/Al_Kuwaitiah_logo.png	14,555
176	200	HTTP	www.alkuwaityah.com	/ArticleImages/430dcec5-c39b-4d13-a8c7-f9108bd1d22e.jpg	20,886
177	200	HTTP	Tunnel to	31.13.74.3:443	0
178	200	HTTP	Tunnel to	31.13.71.21:443	0
179	200	HTTP	Tunnel to	31.13.71.3:443	0
180	200	HTTP	Tunnel to	31.13.71.21:443	0
181	200	HTTP	Tunnel to	31.13.71.21:443	0
182	200	HTTP	Tunnel to	31.13.71.21:443	0
183	200	HTTP	Tunnel to	31.13.71.21:443	0
184	200	HTTP	Tunnel to	31.13.71.21:443	0
185	200	HTTP	Tunnel to	31.13.71.21:443	0
186	200	HTTP	Tunnel to	31.13.71.21:443	0
187	200	HTTP	Tunnel to	31.13.71.21:443	0
188	200	HTTP	sellingportal.alruab...	/UsersWS.asmx/Login?UserName=a&Password=a	61
189	200	HTTP	Tunnel to	31.13.71.3:443	0
190	200	HTTP	sellingportal.alruab...	/UsersWS.asmx/Login?UserName=hussein&Password=abc	61

http://sellingportal.alruabye.net/UsersWS.asmx/Login?UserName=hussein&Password=abc

Request Headers [Raw] [Header Definitions]

GET /UsersWS.asmx/Login?UserName=hussein&Password=abc HTTP/1.1

Client

Accept-Encoding: gzip

User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.2.2; BLU STUDIO 5.5 S Build/JDQ39)

Transport

```
HTTP/1.1 200 OK
Cache-Control: private, max-age=0
Server: Microsoft-IIS/8.5
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Thu, 17 Mar 2016 02:50:00 GMT
Content-Length: 61

{"UserID":0,"Message":" user name or password is in correct"}
```

How do we protect our app data in transit, then?

We should use HTTPS when sending data over the network.

In order to use HTTPS, all we need to do is call **openConnection()** on a URL with "https" in front of it, then cast the resulting connection to HttpsURLConnection.

Here is the HTTP version of our code:

```
public void buloginckic(View view) {
      //get user name and password
      EditText UserName=(EditText)findViewById(R.id.EDTUserName);
      EditText Password=(EditText)findViewById(R.id.EDTpassword);

      // send user name and password over the http
      String url="http://sellingportal.alruabye.net/UsersWS.asmx/Login?UserName="+
UserName.getText().toString() +"&Password="+ Password.getText().toString();

      // start background task
      new MyAsyncTaskGetNews().execute(url, "news");
   }

   // get news from server
   public class MyAsyncTaskGetNews extends AsyncTask<String, String, String> {
      @Override
      protected String doInBackground(String... params) {
        try {
            String NewsData;
            //define the url we have to connect with
            URL url = new URL(params[0]);
            //make connect with url and send request
            HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
            //waiting for 7000ms for response
            urlConnection.setConnectTimeout(7000);//set timeout to 5 seconds
```

The relevant lines are highlighted above – some code has been omitted for clarity.

Unfortunately, the website we have been using (in our example code) to submit our username/password over does not support HTTPS, so we will not be able to demonstrate it in our current code. Below is a code sample of how HTTPS would be accomplished.

```
URL url = new URL("https://en.wikipedia.org/wiki/Main_Page");
HttpsURLConnection urlConnection = (HttpsURLConnection) url.openConnection();
try {
   InputStream in = new BufferedInputStream(urlConnection.getInputStream());
   readStream(in);
} finally {
   urlConnection.disconnect();
}
```

Of course, we would then need to import **javax.net.ssl.HttpsURLConnection** instead of **java.net.HttpURLConnection**.

Resources
For more information, consult these links:

i. HttpURLConnection Android API Reference
ii. Security with HTTPS and SSL