

Seguridad en dispositivos móviles

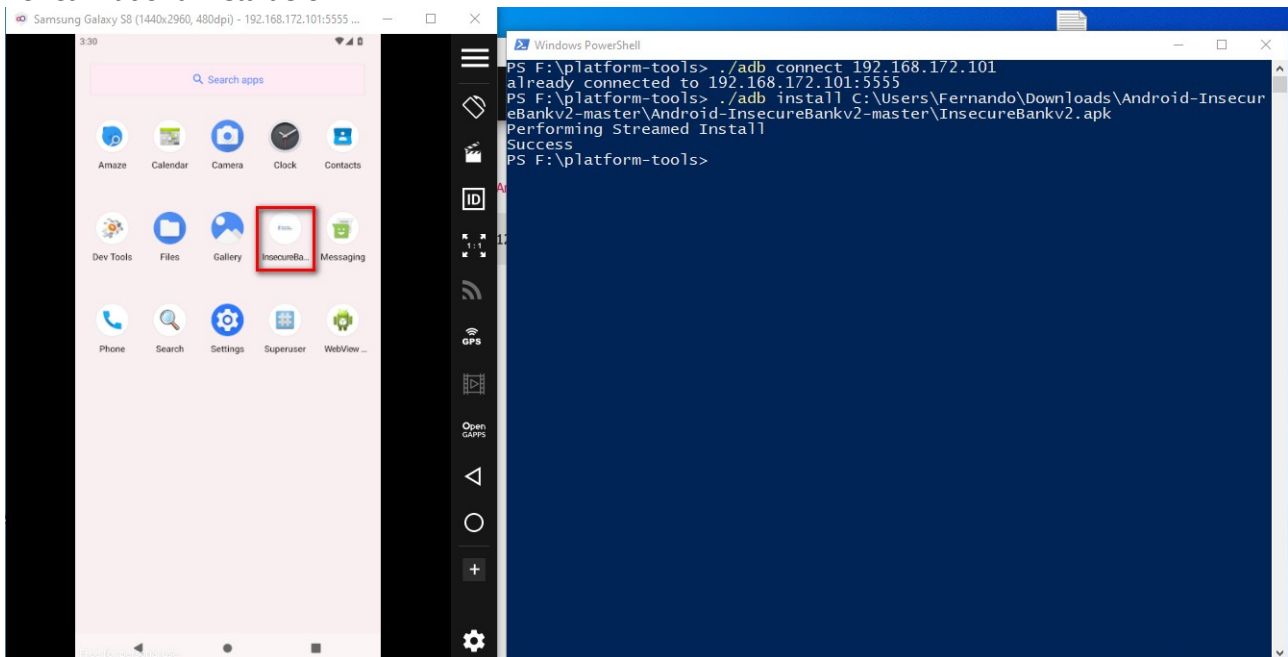
Fernando A. Lorenzo Vázquez

Sumario

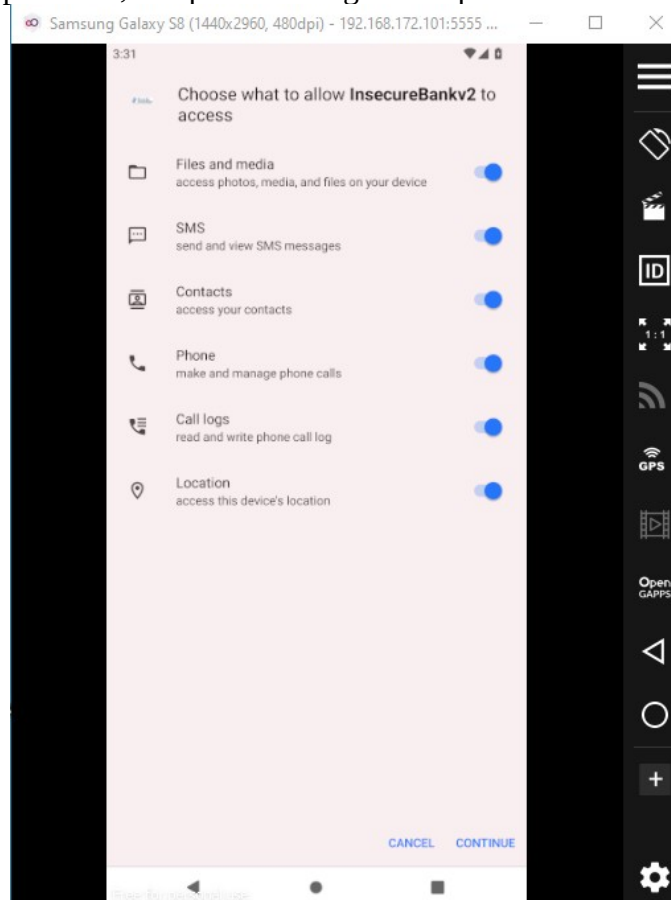
1-Preparación del contorno.....	3
2-Análisis dinámico.....	6
2.1-Vulnerabilidad de inicio de sesión.....	6
2.2-Security Misconfiguration.....	7
2.3-Omisión de Inicio de sesión y de cambio de contraseña.....	8
2.4-Botón de creación de usuario para administradores.....	9
2.5-Login de desarrollador.....	12
2.6-Guardado de contraseñas inseguro.....	13
2.7-Android Backup Enabled.....	16
2.8-Filtrado de información sensible por pantalla.....	17
3-Análisis Dinámico.....	18
3.1-Debug mode enabled.....	18
3.2-Insecure Logs.....	19
3.3-Datos guardados de forma insegura.....	20
3.4-Keyboard Cache.....	23
3.5-Pasteboard Vulnerability.....	23
3.6-Exported Broadcast Receiver.....	24
3.7-Exported TrackUsersProvider.....	25
3.8-Utilización de protocolo de red no seguro (HTTP).....	26
3.9-Vulnerable a ataques de fuerza bruta/diccionario.....	27
3.10-RootDetection Bypass.....	27

1-Preparación del contorno

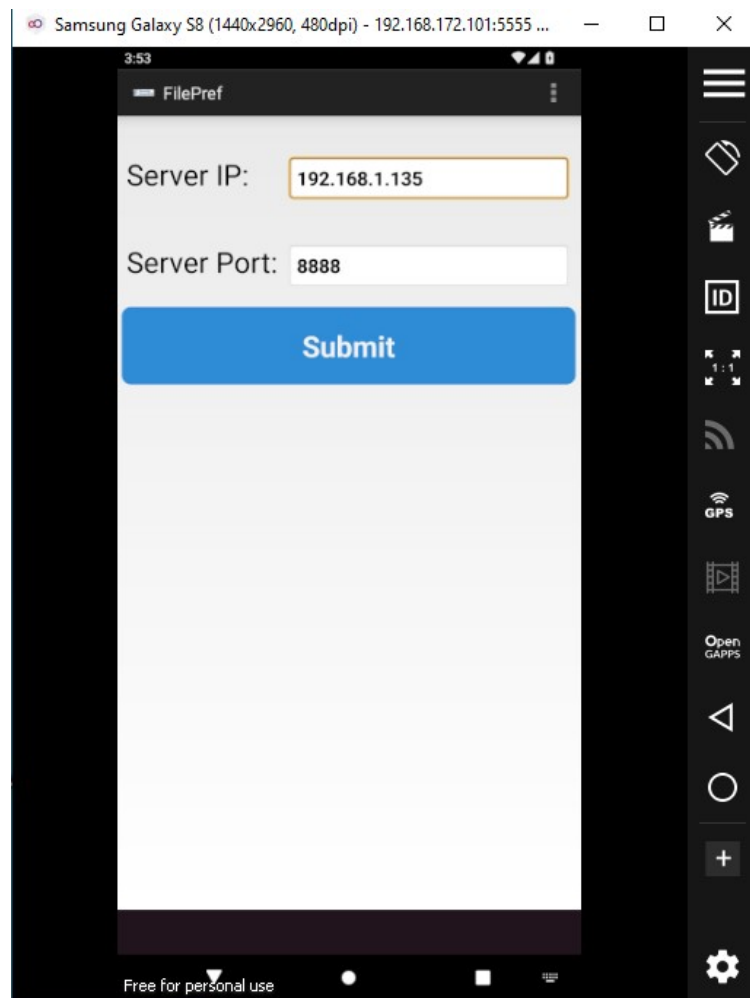
He utilizado Genimotion como emulador android y con la ayuda de Android Debug Bridge (adb), he realizado la instalación



A la hora de iniciar la aplicación, nos pedirá los siguientes permisos



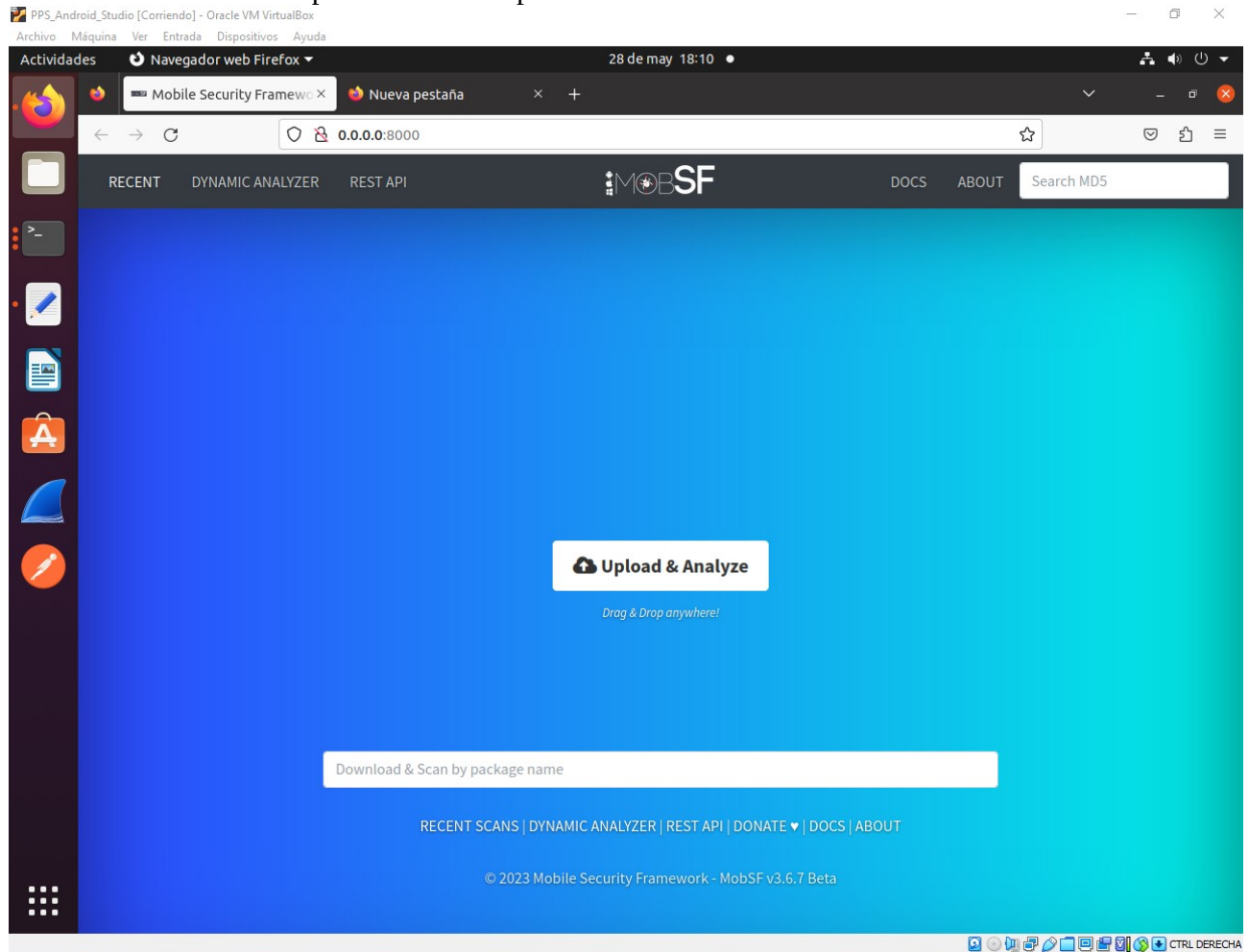
Configuramos la ip y salida de la aplicación



Para realizar el análisis estático, utilizaré [mobsf](#)

```
usuario@pps:~/Escritorio/Mobile-Security-Framework-MobSF$ docker pull opensecurity/mobile-security-framework-mobsf:latest
latest: Pulling from opensecurity/mobile-security-framework-mobsf
ca1778b69356: Pull complete
53763b0312d3: Pull complete
4ef31fb6486a: Pull complete
3c20f21726a6: Pull complete
34bdde6bb9a7: Pull complete
55cd6e25e30d91: Pull complete
07dbccd83c35: Pull complete
1b09e220e573: Pull complete
be5a09ddabb1: Pull complete
58d4d8e13550: Pull complete
725c3f4665be: Pull complete
21bb6c29bf26: Pull complete
389fd1a73610: Pull complete
4f4fb700ef54: Pull complete
26d1833c300e: Pull complete
Digest: sha256:927bd2df2deae205af5e3a30183c9c9e24ceb6698bfe131361790d532490263a
Status: Downloaded newer image for opensecurity/mobile-security-framework-mobsf:latest
docker.io/opensecurity/mobile-security-framework-mobsf:latest
usuario@pps:~/Escritorio/Mobile-Security-Framework-MobSF$ docker run -it --rm -p 8000:8000 opensecurity/mobile-security-framework-mobsf:latest
[INFO] 28/May/2023 15:51:21 -
[MOBSFW]
[INFO] 28/May/2023 15:51:21 - Mobile Security Framework v3.6.7 Beta
REST API Key: 47712d1c2b83cc24eb0d5969d73a4fffd924984b29f2aa1fce201a7a80b108ff
[INFO] 28/May/2023 15:51:21 - OS: Linux
[INFO] 28/May/2023 15:51:21 - Platform: Linux-5.15.0-53-generic-x86_64-with-glibc2.29
[INFO] 28/May/2023 15:51:21 - Dist: ubuntu 20.04 Focal Fossa
[INFO] 28/May/2023 15:51:21 - MobSF Basic Environment Check
[WARNING] 28/May/2023 15:51:21 - Dynamic Analysis related functions will not work.
Make sure a Genymotion Android VM/Android Studio Emulator is running before performing Dynamic Analysis.
No changes detected
[INFO] 28/May/2023 15:51:21 - Checking for Update.
[INFO] 28/May/2023 15:51:22 - No updates available.
```

Ahora arrastraremos el apk a esta nueva ip creada

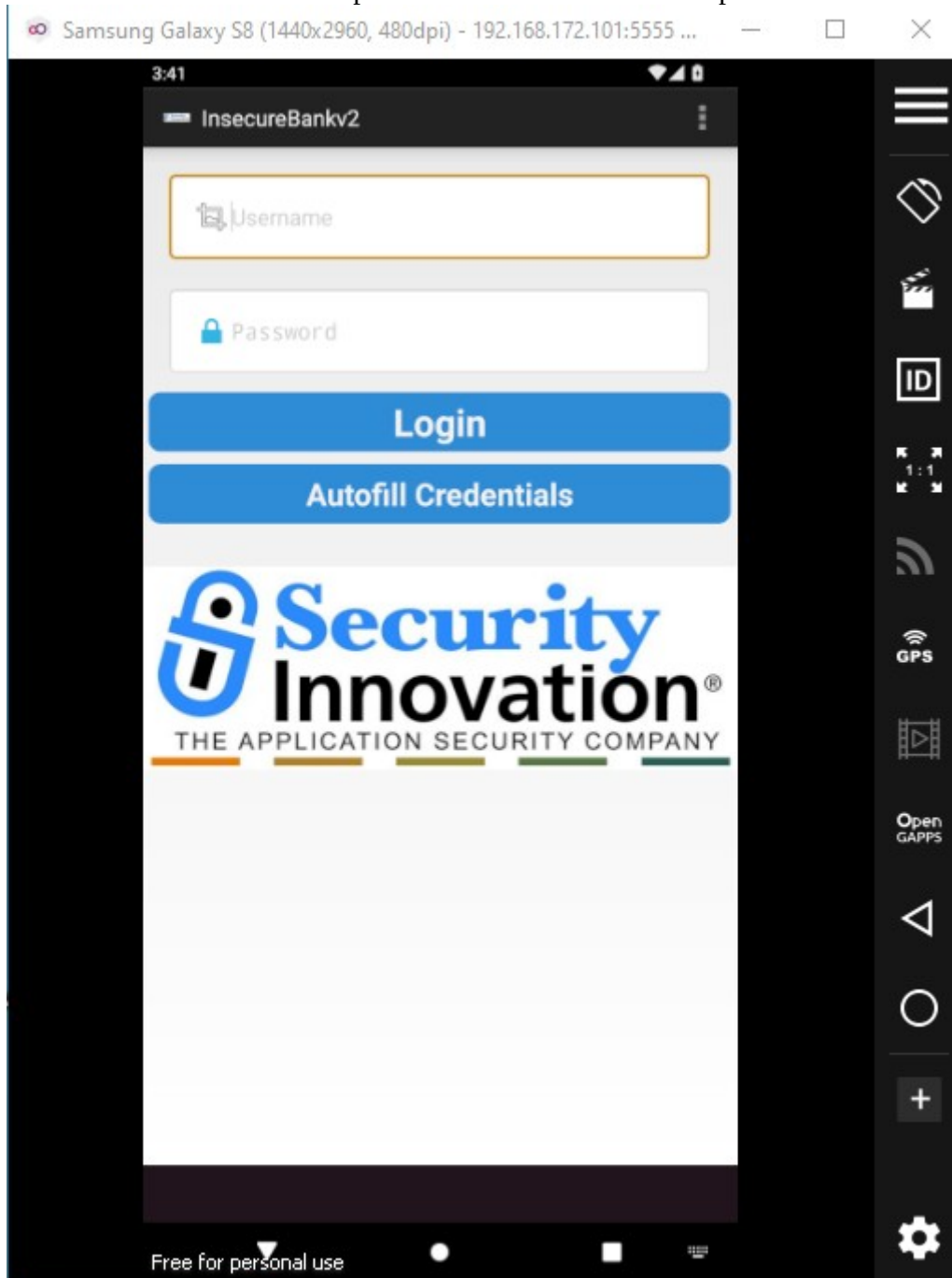


2-Análisis dinámico

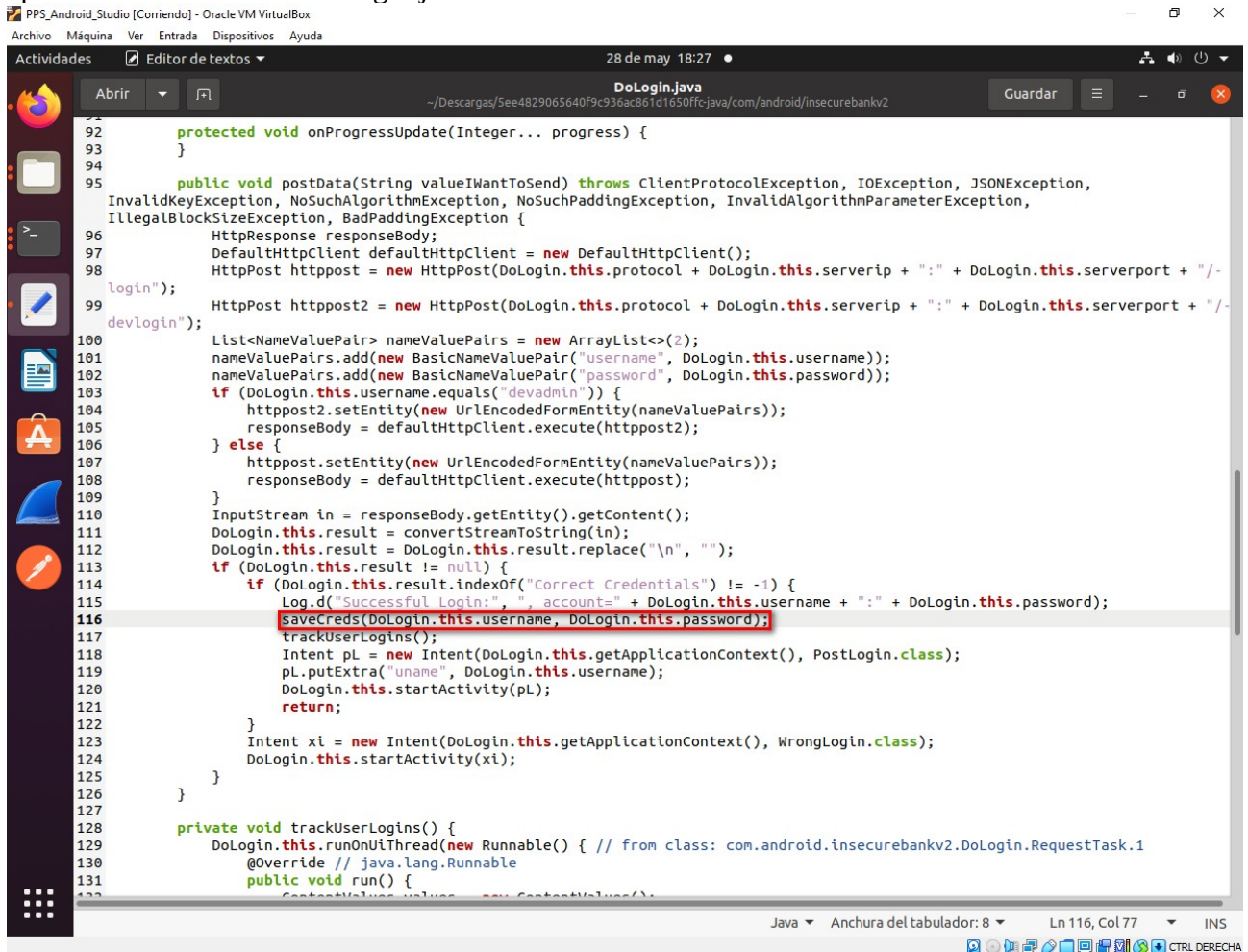
para realizar ingeniería inversa sobre esta aplicación, utilizaré [mbsf](#)

2.1-Vulnerabilidad de inicio de sesión

En la ventana principal podemos observar un boton “autofill Credentials”. Si bien este boton no funciona la primera vez que intentemos iniciar sesión, una vez hayamos iniciado sesión por lo menos una vez, estas credenciales se guardarán las credenciales del ultimo usuario que haya iniciado sesión habilitando el inicio de personas no deseadas en esta aplicación



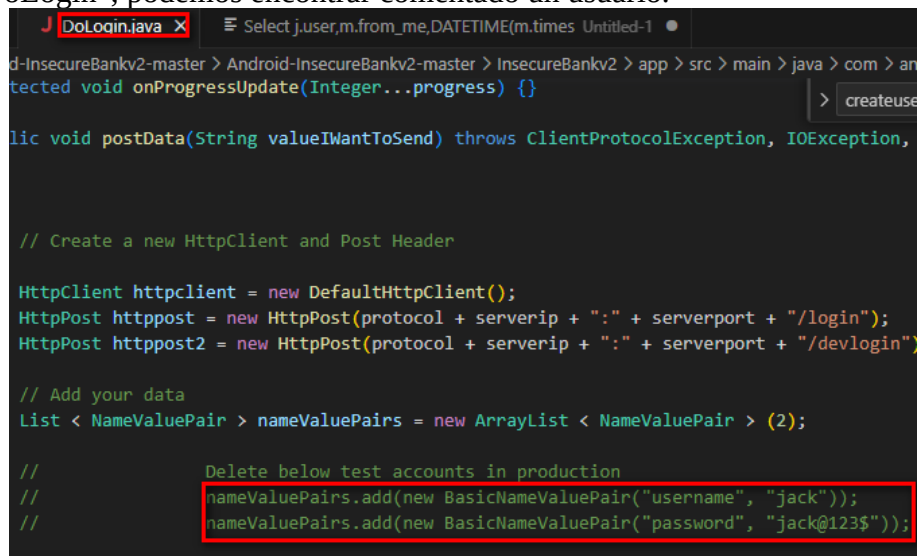
Esta vulnerabilidad se arreglaría suprimiendo la llamada al método “saveCreds” en el método “postData” de la clase “DoLogin.java”.



```
92     protected void onProgressUpdate(Integer... progress) {
93     }
94
95     public void postData(String valueIWantToSend) throws ClientProtocolException, IOException, JSONException,
InvalidKeyException, NoSuchAlgorithmException, NoSuchPaddingException, InvalidAlgorithmParameterException,
IllegalBlockSizeException, BadPaddingException {
96         HttpResponse responseBody;
97         DefaultHttpClient defaultHttpClient = new DefaultHttpClient();
98         HttpPost httpPost = new HttpPost(DoLogin.this.protocol + DoLogin.this.serverip + ":" + DoLogin.this.serverport + "/" +
"login");
99         HttpPost httpPost2 = new HttpPost(DoLogin.this.protocol + DoLogin.this.serverip + ":" + DoLogin.this.serverport + "/" +
"devlogin");
100         List<NameValuePair> nameValuePairs = new ArrayList<>(2);
101         nameValuePairs.add(new BasicNameValuePair("username", DoLogin.this.username));
102         nameValuePairs.add(new BasicNameValuePair("password", DoLogin.this.password));
103         if (DoLogin.this.username.equals("devadmin")) {
104             httpPost2.setEntity(new UrlEncodedFormEntity(nameValuePairs));
105             responseBody = defaultHttpClient.execute(httpPost2);
106         } else {
107             httpPost.setEntity(new UrlEncodedFormEntity(nameValuePairs));
108             responseBody = defaultHttpClient.execute(httpPost);
109         }
110         InputStream in = responseBody.getEntity().getContent();
111         DoLogin.this.result = convertStreamToString(in);
112         DoLogin.this.result = DoLogin.this.result.replace("\n", "");
113         if (DoLogin.this.result != null) {
114             if (DoLogin.this.result.indexOf("Correct Credentials") != -1) {
115                 Log.d("Successful Login:", "account=" + DoLogin.this.username + ":" + DoLogin.this.password);
116                 saveCreds(DoLogin.this.username, DoLogin.this.password);
117                 trackUserLogins();
118                 Intent pL = new Intent(DoLogin.this.getApplicationContext(), PostLogin.class);
119                 pL.putExtra("uname", DoLogin.this.username);
120                 DoLogin.this.startActivity(pL);
121                 return;
122             }
123             Intent xi = new Intent(DoLogin.this.getApplicationContext(), WrongLogin.class);
124             DoLogin.this.startActivity(xi);
125         }
126     }
127
128     private void trackUserLogins() {
129         DoLogin.this.runOnUiThread(new Runnable() { // from class: com.android.insecurebankv2.DoLogin.RequestTask.1
130             @Override // java.lang.Runnable
131             public void run() {
132                 ContactUserLogins();
133             }
134         });
135     }
136 }
```

2.2-Security Misconfiguration

En la clase “DoLogin”, podemos encontrar comentado un usuario.



```
d-InsecureBankv2-master > Android-InsecureBankv2-master > InsecureBankv2 > app > src > main > java > com > and
ected void onProgressUpdate(Integer...progress) {}
> createuse

lic void postData(String valueIWantToSend) throws ClientProtocolException, IOException,

// Create a new HttpClient and Post Header

HttpClient httpclient = new DefaultHttpClient();
HttpPost httpPost = new HttpPost(protocol + serverip + ":" + serverport + "/login");
HttpPost httpPost2 = new HttpPost(protocol + serverip + ":" + serverport + "/devlogin");

// Add your data
List < NameValuePair > nameValuePairs = new ArrayList < NameValuePair > (2);

// Delete below test accounts in production
// nameValuePairs.add(new BasicNameValuePair("username", "jack"));
// nameValuePairs.add(new BasicNameValuePair("password", "jack@123$"));
```

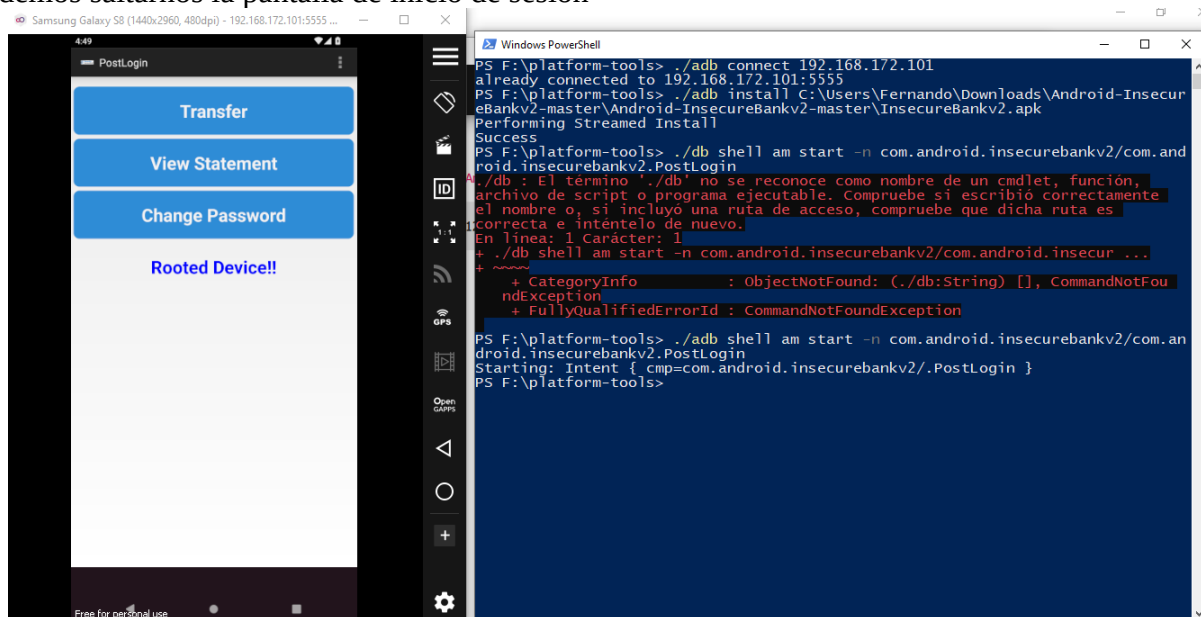
2.3-Omisión de Inicio de sesión y de cambio de contraseña

A parte del código, también podemos acceder al AndroidManifest.xml, donde podemos encontrar las siguientes actividades 'exportadas' (por lo que otras aplicaciones pueden acceder a esta)

```
20.     <intent-filter>
21.         <action android:name="android.intent.action.MAIN" />
22.         <category android:name="android.intent.category.LAUNCHER" />
23.     </intent-filter>
24. </activity>
25. <activity android:label="@string/title_activity_file_pref" android:name="com.android.insecurebankv2.FilePrefActivity" android:windowSoftInputModes="adjustN
26. <activity android:label="@string/title_activity_do_login" android:name="com.android.insecurebankv2.DoLogin" />
27. <activity android:label="@string/title_activity_post_login" android:name="com.android.insecurebankv2.PostLogin" android:exported="true" />
28. <activity android:label="@string/title_activity_wrong_login" android:name="com.android.insecurebankv2.WrongLogin" />
29. <activity android:label="@string/title_activity_do_transfer" android:name="com.android.insecurebankv2.DoTransfer" android:exported="true" />
30. <activity android:label="@string/title_activity_view_statement" android:name="com.android.insecurebankv2.ViewStatement" android:exported="true" />
31. <provider android:name="com.android.insecurebankv2.TrackUserContentProvider" android:exported="true" android:authorities="com.android.insecurebankv2.Track
32. <receiver android:name="com.android.insecurebankv2.MyBroadcastReceiver" android:exported="true">
33.     <intent-filter>
34.         <action android:name="theBroadcast" />
35.     </intent-filter>
36. </receiver>
37. <activity android:label="@string/title_activity_change_password" android:name="com.android.insecurebankv2.ChangePassword" android:exported="true" />
38. <activity android:theme="@android:style/Theme.Translucent" android:name="com.google.android.gms.ads.AdActivity" android:configChanges="keyboard|keyboardHic
39. <activity android:theme="@style/Theme.IAPTheme" android:name="com.google.android.gms.ads.purchase.InAppPurchaseActivity" />
40. <meta-data android:name="com.google.android.gms.version" android:value="@integer/google_play_services_version" />
41. <meta-data android:name="com.google.android.gms.wallet.api.enabled" android:value="true" />
42. <receiver android:name="com.google.android.gms.wallet.EnableWalletOptimizationReceiver" android:exported="false">
43.     <intent-filter>
44.         <action android:name="com.google.android.gms.wallet.ENABLE_WALLET_OPTIMIZATION" />
45.     </intent-filter>
46. </receiver>
47. </application>
48. </manifest>
```

Para explotar esta vulnerabilidad debemos volver a nuestra terminal, donde gracias al comando:
adb shell am start -n
com.android.insecurebankv2/com.android.insecurebankv2.PostLogin

Podemos saltarnos la pantalla de inicio de sesión



Para solucionar este error, debemos establecer el valor de estas actividades críticas a false


```

*AndroidManifest.xml: Bloc de notas
Archivo Edición Formato Ver Ayuda

    android:name=".LoginActivity"
    android:label="@string/app_name" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity
    android:name=".FilePrefActivity"
    android:label="@string/title_activity_file_pref"
    android:windowSoftInputMode="stateVisible|adjustResize|adjustPan">
</activity>
<activity
    android:name=".DoLogin"
    android:label="@string/title_activity_do_login" >
</activity>
<activity
    android:name=".PostLogin"
    android:exported="false"
    android:label="@string/title_activity_post_login" >
</activity>
<activity
    android:name=".WrongLogin"
    android:label="@string/title_activity_wrong_login" >
</activity>
<activity
    android:name=".DoTransfer"
    android:exported="false"
    android:label="@string/title_activity_do_transfer" >
</activity>
<activity
    android:name=".ViewStatement"
    android:exported="false"
    android:label="@string/title_activity_view_statement" >
</activity>
<provider
    android:name=".TrackUserContentProvider"
    android:authorities="com.android.insecurebankv2.TrackUserContentProvider"
    android:exported="true" >
</provider>
<receiver
    android:name=".MyBroadCastReceiver"
    android:exported="true" >
    <intent-filter>
        <action android:name="theBroadcast" >
        </action>
    </intent-filter>

```

2.4-Botón de creación de usuario para administradores

En la clase “LoginActivity.java”, podemos encontrar una comprobación de usuario administrador, si el usuario resulta ser administrador, se hará visible un botón “CreateUser”, en caso contrario, permanecerá desaparecido.

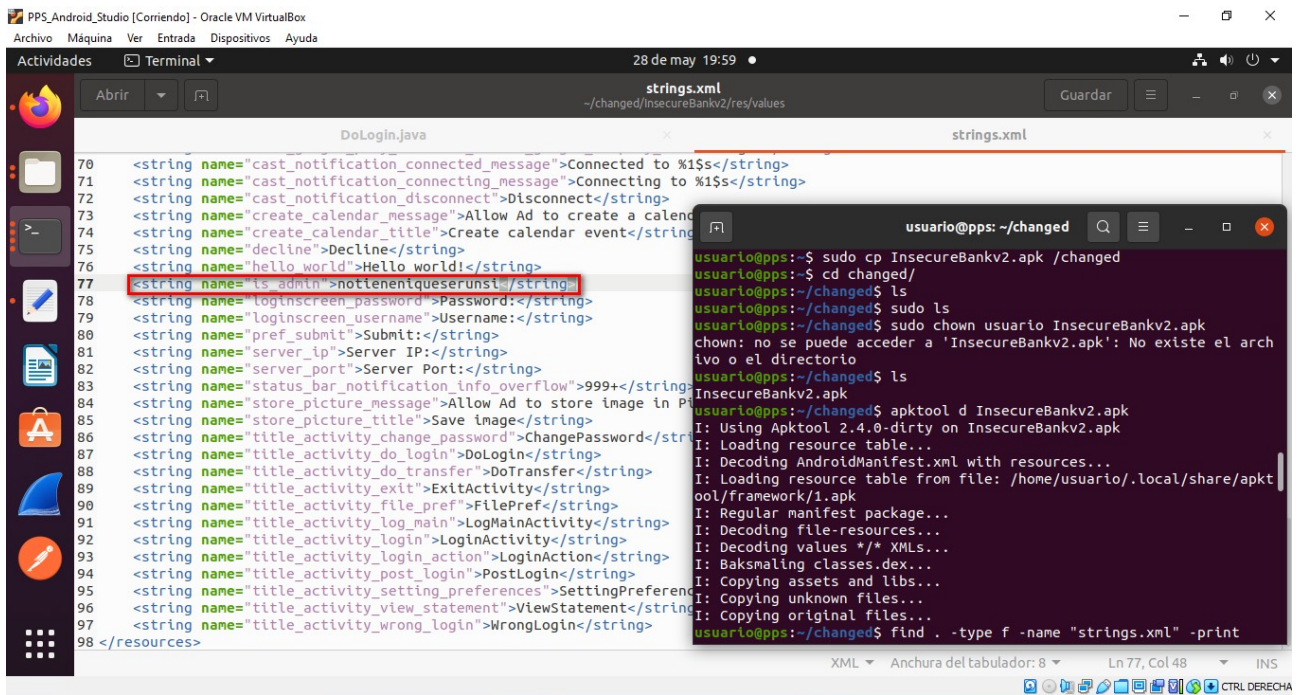
```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_log_main);
    String mess = getResources().getString(R.string.is_admin);
    if (mess.equals("no")) {
        View button_CreateUser = findViewById(R.id.button_CreateUser);
        button_CreateUser.setVisibility(View.GONE);
    }
    login_buttons = (Button) findViewById(R.id.login_button);

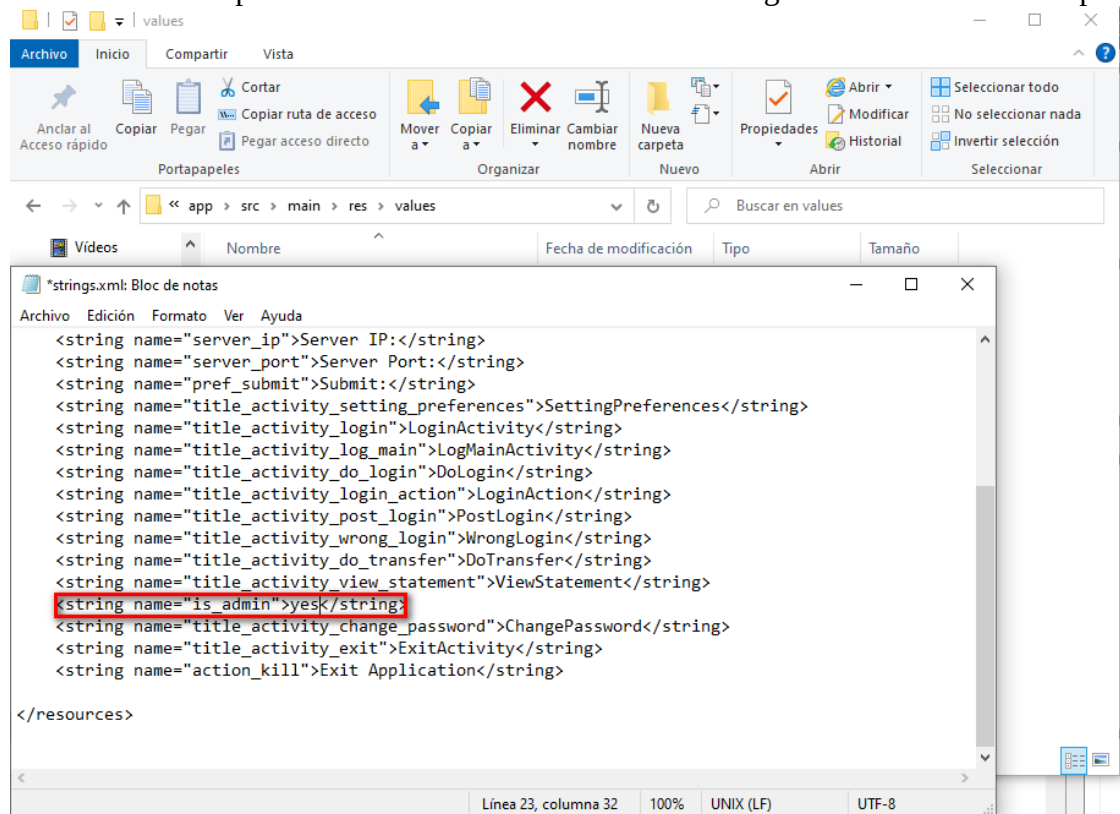
```

Normalmente, esto no supone un problema, pues a la hora de iniciar sesión nunca podremos ser administradores.

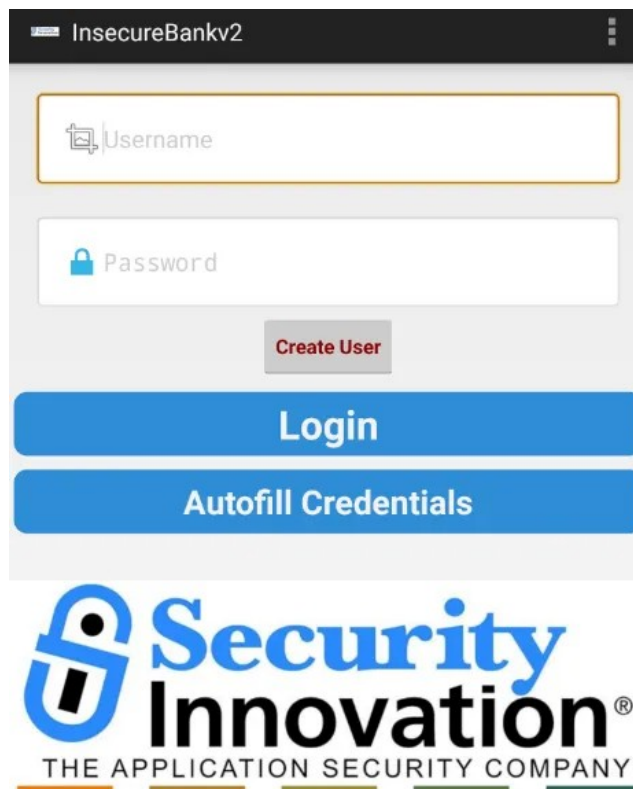
Sin embargo, esta comprobación está basada en un archivo de recursos con una propiedad “is_admin”, por lo tanto podemos descompilar nuestra apk, cambiar este valor y volver a montarla.



Este cambio también se puede realizar directamente desde el código sin necesidad de este proceso



Si recompilamos nuestra aplicación con ‘apktool b -f -d InsecureBankv2/‘ tendremos acceso a este nuevo botón



Sin embargo este boton no tiene ninguna razón de existencia más allá de mostrar la existencia de la vulnerabilidad, pues si nos dirigimos a su código, este no tiene utilidad

```
/*
The function that allows the user to create new user credentials.
This functionality is available only to the admin user.
<<WIP Code>>
ToDo: Add functionality here.
*/
protected void createUser() {
    Toasteroid.show(this, "Create User functionality is still Work-In-Progress!!", Toasteroid.STYLES.WARNING, Toasteroid.LENGTH_LONG);
}
```

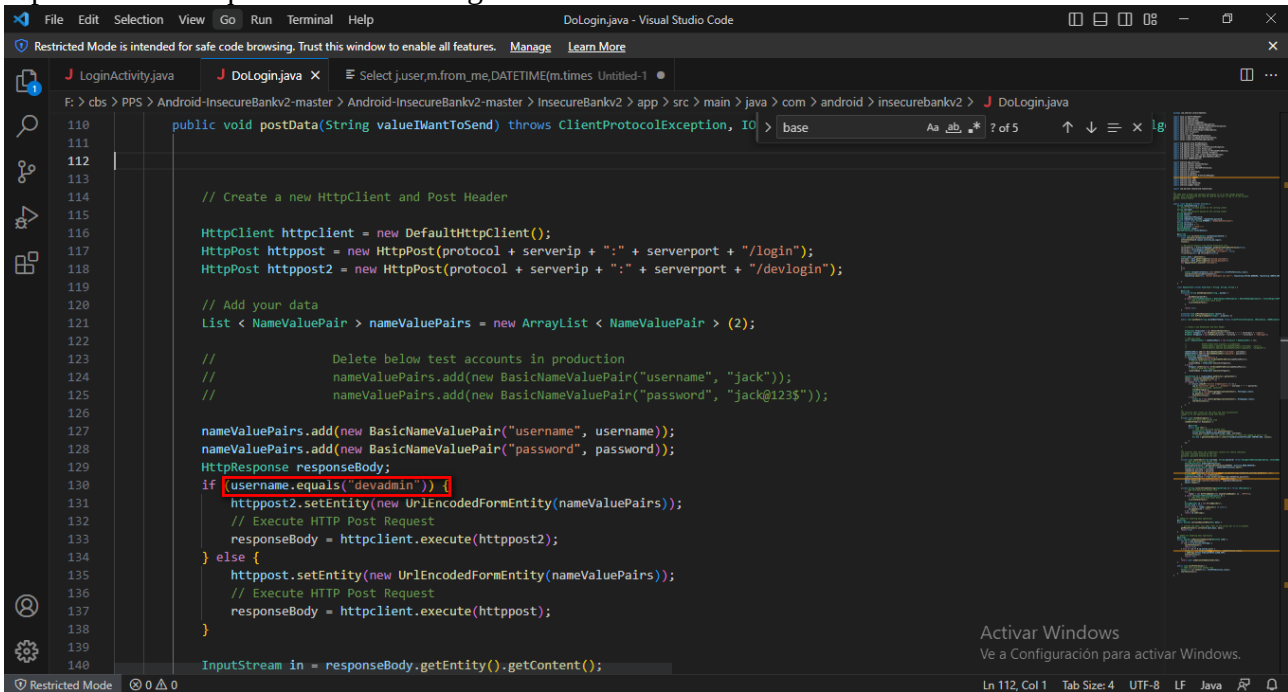
Podemos arreglar esta vulnerabilidad cambiando el código de comprobación para hacerse visible y evitar así el acceso a este botón.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_log_main);
    String mess = getResources().getString(R.string.is_admin);
    View button_CreateUser = findViewById(R.id.button_CreateUser);
    button_CreateUser.setVisibility(View.GONE);
}
```

claro que si no vamos a mostrar este botón en ningún caso la mejor solución sería la eliminación completa de este, así como del método “createUser” solo accesible mediante este botón (Aunque no mostrada, he realizado esta corrección en el código final).

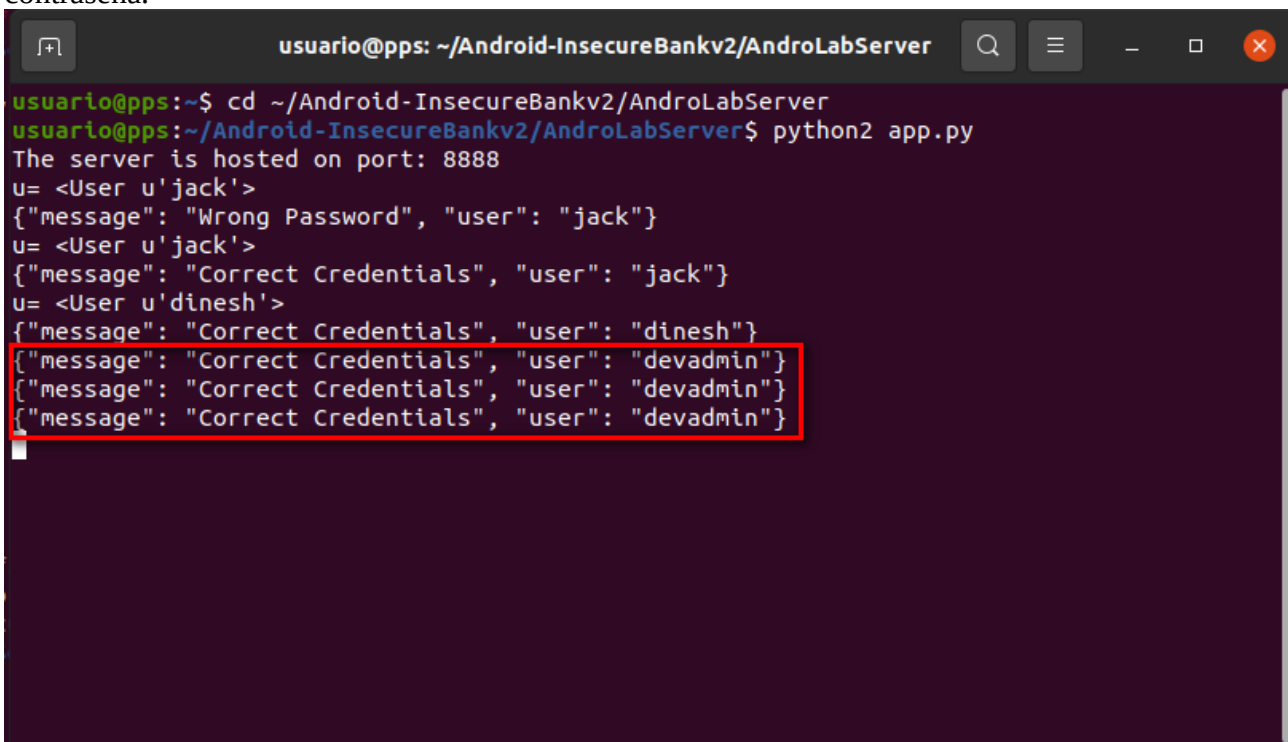
2.5-Login de desarrollador

En el método “postData” de la clase “DoLogin”, podemos ver que existen diferencias entre un usuario normal así como un usuario desarrollador, además de descubrir la inexistencia de usuarios con el rol de desarrolladores, en su lugar únicamente existe un usuario desarrollador y se comprueba específicamente que el nombre sea igual a este usuario.



```
110 public void postData(String valueIWantToSend) throws ClientProtocolException, IOException {
111
112
113
114     // Create a new HttpClient and Post Header
115
116     HttpClient httpClient = new DefaultHttpClient();
117     HttpPost httpPost = new HttpPost(protocol + serverip + ":" + serverport + "/login");
118     HttpPost httpPost2 = new HttpPost(protocol + serverip + ":" + serverport + "/devlogin");
119
120     // Add your data
121     List<NameValuePair> nameValuePairs = new ArrayList<NameValuePair> (2);
122
123     // Delete below test accounts in production
124     // nameValuePairs.add(new BasicNameValuePair("username", "jack"));
125     // nameValuePairs.add(new BasicNameValuePair("password", "jack@123$"));
126
127     nameValuePairs.add(new BasicNameValuePair("username", username));
128     nameValuePairs.add(new BasicNameValuePair("password", password));
129     HttpResponse responseBody;
130     if (username.equals("devadmin")) {
131         httpPost2.setEntity(new UrlEncodedFormEntity(nameValuePairs));
132         // Execute HTTP Post Request
133         responseBody = httpClient.execute(httpPost2);
134     } else {
135         httpPost.setEntity(new UrlEncodedFormEntity(nameValuePairs));
136         // Execute HTTP Post Request
137         responseBody = httpClient.execute(httpPost);
138     }
139
140     InputStream in = responseBody.getEntity().getContent();
```

Podemos intentar iniciar sesión con este usuario y podremos iniciar sesión sin necesidad de contraseña.



```
usuario@pps: ~/Android-InsecureBankv2/AndroLabServer
usuario@pps:~/Android-InsecureBankv2/AndroLabServer$ python2 app.py
The server is hosted on port: 8888
u= <User u'jack'>
{"message": "Wrong Password", "user": "jack"}
u= <User u'jack'>
{"message": "Correct Credentials", "user": "jack"}
u= <User u'dinesh'>
{"message": "Correct Credentials", "user": "dinesh"}
{"message": "Correct Credentials", "user": "devadmin"}
{"message": "Correct Credentials", "user": "devadmin"}
{"message": "Correct Credentials", "user": "devadmin"}
```

La mejor solución a esta vulnerabilidad, sería agregar nuevos campos a los usuarios, entre ellos una propiedad “tipo” en la que se pudiese especificar si fuese desarrollador o no, de esta forma no sería necesaria esta comprobación.

Como no existe ninguna clase en la que se definan los usuarios, una solución temporal puede ser la eliminación de esta comprobación

```
nameValuePairs.add(new BasicNameValuePair("username", username));
nameValuePairs.add(new BasicNameValuePair("password", password));
HttpResponse responseBody;
httpPost.setEntity(new UrlEncodedFormEntity(nameValuePairs));
// Execute HTTP Post Request
responseBody = httpClient.execute(httpPost);
```

Con este cambio se pierde la funcionalidad de que el usuario desarrollador guarde los datos en un directorio a parte, pero es una solución temporal mientras se trabaja en la creación de una clase “Usuarios” bien formada.

Otra solución temporal podría ser cambiar la contraseña a este usuario para que necesite una, seguiría sin ser una solución definitiva, ya que el nombre de este administrador sigue apareciendo en el código, pero temporalmente puede funcionar.

2.6-Guardado de contraseñas inseguro

Como mostré en el primer punto de esta parte del análisis, usuario y contraseña del último usuario se guardan en un archivo “SharedPreferences” encriptadas con base64 y AES respectivamente.

```
private void saveCreds(String username, String password) throws UnsupportedEncodingException, InvalidKeyException, NoSuchAlgorithmException {
    // TODO Auto-generated method stub
    SharedPreferences mySharedPreferences;
    mySharedPreferences = getSharedPreferences(MYPREFS, Activity.MODE_PRIVATE);
    SharedPreferences.Editor editor = mySharedPreferences.edit();
    rememberme_username = username;
    rememberme_password = password;
    String base64Username = new String(Base64.encodeToString(rememberme_username.getBytes(), 4));
    CryptoClass crypt = new CryptoClass();
    superSecurePassword = crypt.aesEncryptedString(rememberme_password);
    editor.putString("EncryptedUsername", base64Username);
    editor.putString("superSecurePassword", superSecurePassword);
    editor.commit();
}
```

En la clase “LoginActivity” podemos encontrar el método ‘fillData()’ que obtiene las credenciales guardadas y las desencripta manualmente mediante un método.

```
protected void fillData() throws UnsupportedEncodingException, InvalidKeyException, NoSuchAlgorithmException, NoSuchPaddingException, InvalidAlgorithmParameterException {
    // TODO Auto-generated method stub
    SharedPreferences settings = getSharedPreferences(MYPREFS, 0);
    final String username = settings.getString("EncryptedUsername", null);
    final String password = settings.getString("superSecurePassword", null);

    if(username!=null && password!=null)
    {
        byte[] usernameBase64Byte = Base64.decode(username, Base64.DEFAULT);
        try {
            usernameBase64ByteString = new String(usernameBase64Byte, "UTF-8");
        } catch (UnsupportedEncodingException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        Username_Text = (EditText) findViewById(R.id.loginscreen_username);
        Password_Text = (EditText) findViewById(R.id.loginscreen_password);
        Username_Text.setText(usernameBase64ByteString);
        CryptoClass crypt = new CryptoClass();
        String decryptedPassword = crypt.aesDecryptedString(password);
        Password_Text.setText(decryptedPassword);
    }
}
```

Para explotar esta vulnerabilidad, podemos movernos a adb para crear una shell y encontrar la ubicación del archivo.

```
PS F:\platform-tools> ./adb shell
vbox86p:/ # cd data/data
vbox86p:/data/data # cd com.android.insecurebankv2/shared_prefs/
vbox86p:/data/data/com.android.insecurebankv2/shared_prefs # ls
com.android.insecurebankv2_preferences.xml  mySharedPreferences.xml
```

Una vez tengamos el archivo localizado, podemos descargarlo con adb pull

```
127|vbox86p:/data/data/com.android.insecurebankv2/shared_prefs # exit
PS F:\platform-tools> ./adb pull /data/data/com.android.insecurebankv2/shared_prefs/mySharedPreferences.xml
PS F:\platform-tools> ./adb pull /data/data/com.android.insecurebankv2/shared_prefs/mySharedPreferences.xml: 1 file pulled, 0 skipped. 0.0 MB/s (225 bytes in 0.013s)
```

Dentro de este archivo tendremos las contraseñas y podemos descifrarlas con cualquier aplicación web



```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
- <map>
  <string name="superSecurePassword">v/sJpihDCo2ckDmLW5Uwiw== </string>
  <string name="EncryptedUsername">amFjaw== </string>
</map>
```

Decode from Base64 format

Simply enter your data then push the decode button.

amFjaw==

For encoded binaries (like images, documents, ...)

UTF-8 Source character set.

☐ Decode each line separately (useful for when you have multiple lines of data)

☒ Live mode OFF Decodes in real-time as you type

DECODE Decodes your data into the plain text

jack

Para la contraseña resulta un poco más complicado, ya que al utilizar cifrado AES, necesitamos conocer la llave.

Sin embargo la podemos encontrar en la clase “CryptoClass”

```
public class CryptoClass {  
  
    // The super secret key used by the encryption function  
    String key = "This is the super secret key 123";  
  
    // The initialization vector used by the encryption function  
    byte[] ivBytes = {  
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00  
    };  
    String plainText;  
    byte[] cipherData;  
    String base64Text;  
    String cipherText;  
}
```

y con estos datos podemos descifrar la contraseña.

The screenshot shows a web-based AES Decrypt tool. On the left, under the 'Recipe' tab, the 'Key' is set to 'This is the super secret key 1...', the 'IV' is '00 00 00 00 00 00 00 00 00 00 00 00', the 'Mode' is 'CBC', and the 'Input' is 'Hex'. The 'Output' is set to 'Raw'. On the right, the 'Input' field contains the hex string 'bf fb 09 a6 28 43 0a 8d 9c 90 39 8b 5b 95 30 8b'. Below this, the 'Output' field displays the decrypted result 'Jack@123\$'. Metadata for the input shows a start and end of 47, a length of 47, and 1 line. The output metadata shows a time of 0ms, a length of 9, and 1 line.

La solución sería cambiar el algoritmo a un algoritmo de cifrado más potente como puede ser SHA512 o SHA256 y añadir un salado para evitar coincidencias con bases de datos con contraseñas guardadas.

Sin embargo, como expliqué en anteriormente, estas comprobaciones resultan inútiles si simplemente presionando un boton se puede iniciar sesión con estas credenciales sin darle ninguna opción al usuario sobre si guardarlas o no, por lo que la solución mostrada anteriormente se puede aplicar en esta hasta cierto punto.

```
InputStream in = responseBody.getEntity().getContent();  
result = convertStreamToString( in );  
result = result.replace("\n", "");  
if (result != null) {  
    if (result.indexOf("Correct Credentials") != -1) {  
        Log.d("Successful Login:", " ", account=" + username + ":" + password);  
        //saveCreds(username, password);  
        trackUserLogins();  
        Intent pL = new Intent(getApplicationContext(), PostLogin.class);  
        pL.putExtra("uname", username);  
        startActivity(pL);  
    } else {  
        Intent xi = new Intent(getApplicationContext(), WrongLogin.class);  
        startActivity(xi);  
    }  
}
```

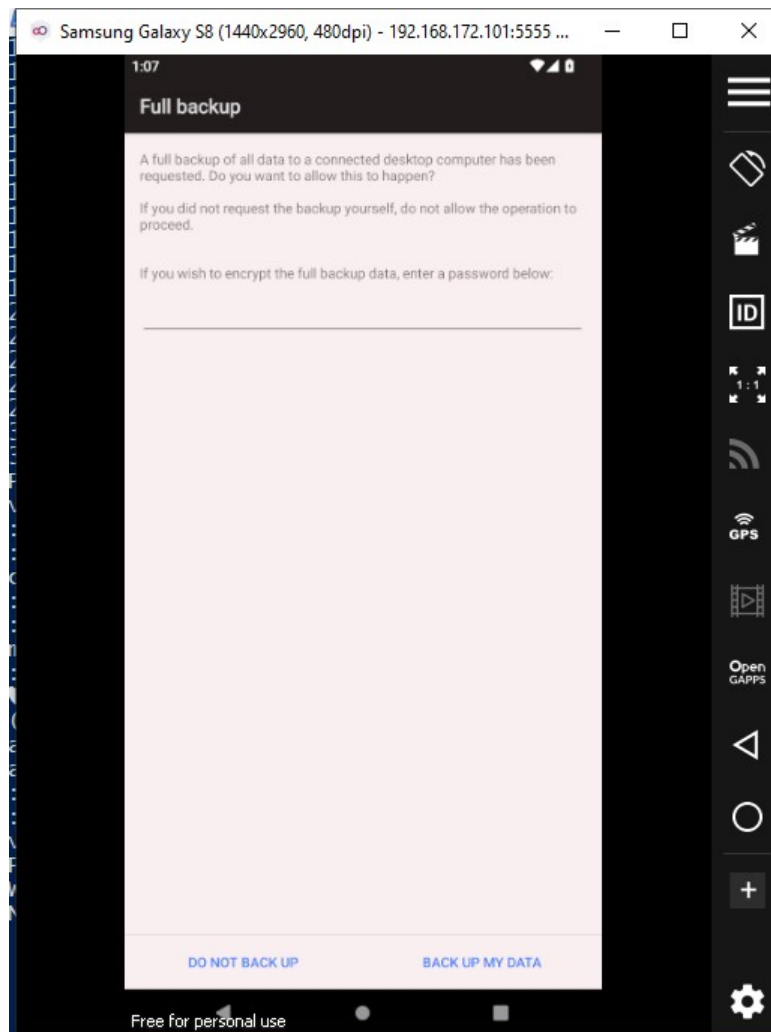
2.7-Android Backup Enabled

Volviendo a nuestro android manifest, podemos ver que, entre otras cosas, tenemos habilitado el modo debug

```
<?xml version="1.0" encoding="UTF-8"?>
- <manifest package="com.android.insecurebankv2" xmlns:android="http://schemas.android.com/apk/res/android">
  <uses-permission android:name="android.permission.INTERNET"/>
  <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
  <uses-permission android:name="android.permission.SEND_SMS"/>
  <!-- To retrieve OAuth 2.0 tokens or invalidate tokens to disconnect a user. This disconnect option is required to comply with the Google+ Sign-in developer policies -->
  <uses-permission android:name="android.permission.USE_CREDENTIALS"/>
  <!-- To retrieve the account name (email) as part of sign-in: -->
  <uses-permission android:name="android.permission.GET_ACCOUNTS"/>
  <!-- To auto-complete the email text field in the login form with the user's emails -->
  <uses-permission android:name="android.permission.READ_PROFILE"/>
  <uses-permission android:name="android.permission.READ_CONTACTS"/>
  <android:uses-permission android:name="android.permission.READ_PHONE_STATE"/>
  <android:uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" android:maxSdkVersion="18"/>
  <android:uses-permission android:name="android.permission.READ_CALL_LOG"/>
- <application android:theme="@android:style/Theme.Holo.Light.DarkActionBar" android:label="@string/app_name" android:icon="@mipmap/ic_launcher"
  android:allowBackup="true">
```

Sabiendo esto podemos realizar un backup de la aplicación con abd backup

```
PS F:\platform-tools> ./adb backup com.android.insecurebankv2
WARNING: adb backup is deprecated and may be removed in a future release
Now unlock your device and confirm the backup operation...
PS F:\platform-tools>
```



Podríamos seguir el proceso de conversión y extracción del tar generado a partir de este

Unidad de USB (F:) > platform-tools		Buscar en platform-tools		
Nombre		Fecha de modificación	Tipo	Tamaño
adb.exe		12/04/2023 17:08	Aplicación	5.881 KB
AdbWinApi.dll		12/04/2023 17:08	Extensión de la ap...	96 KB
AdbWinUsbApi.dll		12/04/2023 17:08	Extensión de la ap...	62 KB
backup.ab		29/05/2023 3:07	Archivo AB	2 KB
dmtracedump.exe		12/04/2023 17:08	Aplicación	236 KB

La mitigación es cambiar “allowBackup” a false

```
<application
    android:allowBackup="false"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:theme="@android:style/Theme.Holo.Light.DarkActionBar">
<!--
    android:theme="@style/AppTheme"-->
```

2.8-Filtrado de información sensible por pantalla

En la clase “DoTransfer”, en el método “run()”, podemos encontrar el siguiente código:

```
@Override
public void run() {
    // TODO Auto-generated method stub
    AsyncHttpTransferPost("result");
    if (result != null) {
        if (result.indexOf("Success") != -1) {
            Toast.show(DoTransfer.this, "Transfer Successful!", Toast.LENGTH_SHORT);

            try {
                JSONObject = new JSONObject(result);
                acc1 = JSONObject.getString("from");
                acc2 = JSONObject.getString("to");
                System.out.println("Message: " + JSONObject.getString("message") + " From: " + from.getText().toString() + " To: " + to.getText().toString() + " Amount: " + amount.getText().toString());
                final String status = new String("VMessage: " + "Success" + " From: " + from.getText().toString() + " To: " + to.getText().toString() + " Amount: " + amount.getText().toString());
            } catch (JSONException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        } else {
            Toast.show(DoTransfer.this, "Transfer Failed!!", Toast.LENGTH_SHORT);
        }
    }
}
```

“System.out.println()” es utilizado para sacar mensajes por consola, por lo que esta linea está filtrando información del número de cuentas.

Para arreglar esta vulnerabilidad, debemos eliminar esta linea de código, de esta forma no habrá filtrado.

```
@Override
public void run() {
    // TODO Auto-generated method stub
    AsyncHttpTransferPost("result");
    if (result != null) {
        if (result.indexOf("Success") != -1) {
            Toast.show(DoTransfer.this, "Transfer Successful!", Toast.LENGTH_SHORT);

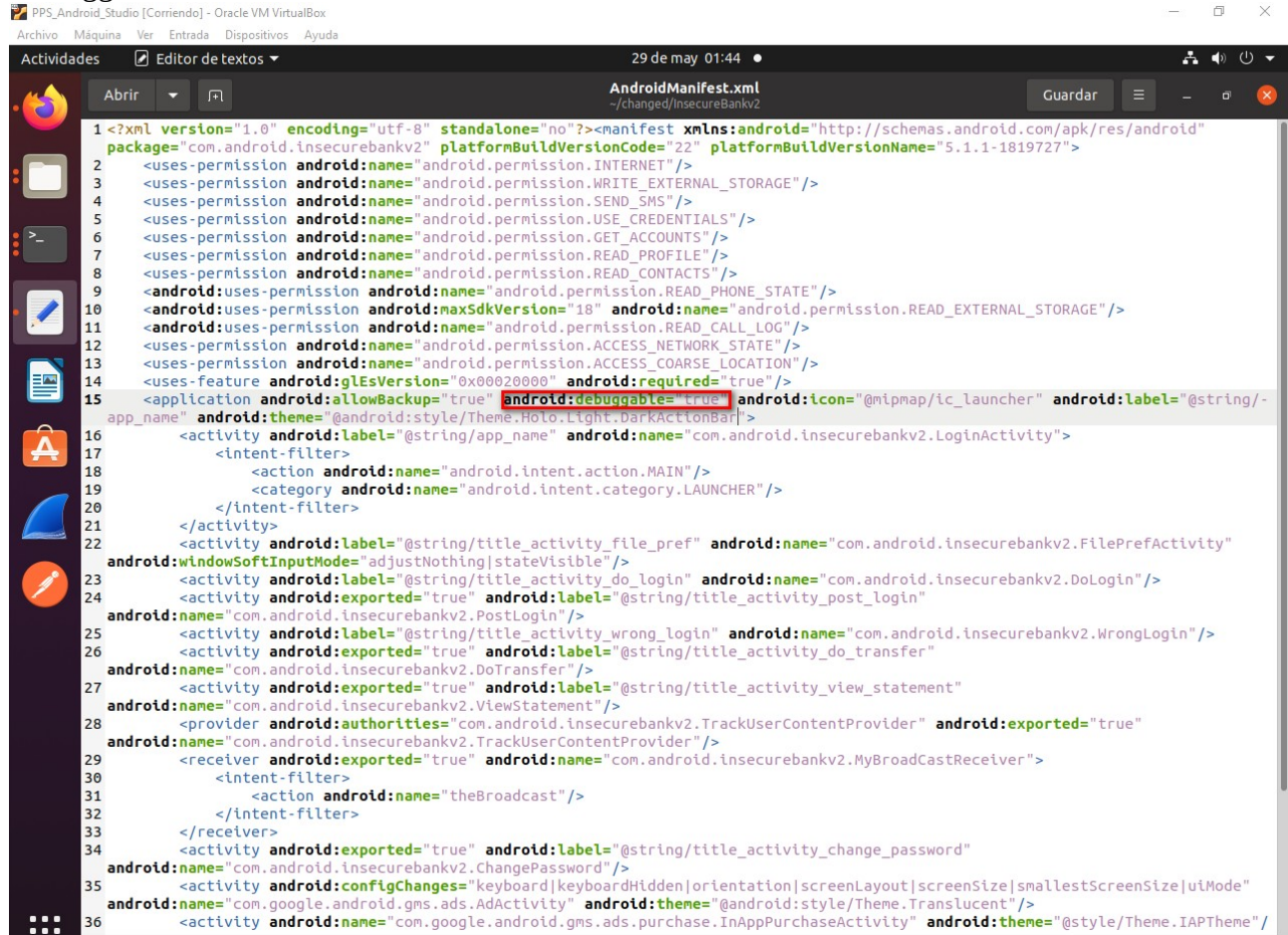
            try {
                JSONObject = new JSONObject(result);
                acc1 = JSONObject.getString("from");
                acc2 = JSONObject.getString("to");

                final String status = new String("VMessage: " + "Success" + " From: " + from.getText().toString() + " To: " + to.getText().toString() + " Amount: " + amount.getText().toString());
            } catch (JSONException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        } else {
            Toast.show(DoTransfer.this, "Transfer Failed!!", Toast.LENGTH_SHORT);
        }
    }
}
```

3-Análisis Dinámico

3.1-Debug mode enabled

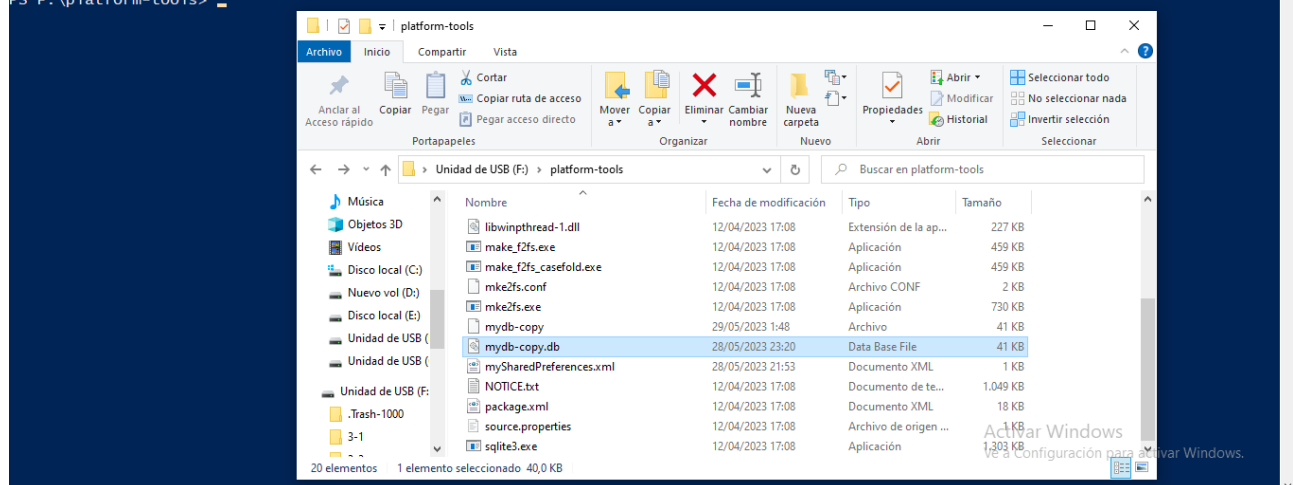
Si miramos el Manifest obtenido gracias a apktool, podemos ver que tenemos la aplicación en modo 'debuggable'



```
1 <?xml version="1.0" encoding="utf-8" standalone="no"?><manifest xmlns:android="http://schemas.android.com/apk/res/android"
2 package="com.android.insecurebankv2" platformBuildVersionCode="22" platformBuildVersionName="5.1.1-1819727">
3   <uses-permission android:name="android.permission.INTERNET"/>
4   <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
5   <uses-permission android:name="android.permission.SEND_SMS"/>
6   <uses-permission android:name="android.permission.USE_CREDENTIALS"/>
7   <uses-permission android:name="android.permission.GET_ACCOUNTS"/>
8   <uses-permission android:name="android.permission.READ_PROFILE"/>
9   <uses-permission android:name="android.permission.READ_CONTACTS"/>
10  <android:uses-permission android:name="android.permission.READ_PHONE_STATE"/>
11  <android:uses-permission android:maxSdkVersion="18" android:name="android.permission.READ_EXTERNAL_STORAGE"/>
12  <android:uses-permission android:name="android.permission.READ_CALL_LOG"/>
13  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
14  <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
15  <uses-feature android:glEsVersion="0x00020000" android:required="true"/>
16  <application android:allowBackup="true" android:debuggable="true" android:icon="@mipmap/ic_launcher" android:label="@string/-
17    app_name" android:theme="@android:style/Theme.Holo.Light.DarkActionBar">
18    <activity android:label="@string/app_name" android:name="com.android.insecurebankv2.LoginActivity">
19      <intent-filter>
20        <action android:name="android.intent.action.MAIN"/>
21        <category android:name="android.intent.category.LAUNCHER"/>
22      </intent-filter>
23    </activity>
24    <activity android:label="@string/title_activity_file_pref" android:name="com.android.insecurebankv2.FilePrefActivity"
25      android:windowSoftInputMode="adjustNothing|stateVisible"/>
26    <activity android:label="@string/title_activity_do_login" android:name="com.android.insecurebankv2.DoLogin"/>
27    <activity android:exported="true" android:label="@string/title_activity_post_login"
28      android:name="com.android.insecurebankv2.PostLogin"/>
29    <activity android:label="@string/title_activity_wrong_login" android:name="com.android.insecurebankv2.WrongLogin"/>
30    <activity android:exported="true" android:label="@string/title_activity_do_transfer"
31      android:name="com.android.insecurebankv2.DoTransfer"/>
32    <activity android:exported="true" android:label="@string/title_activity_view_statement"
33      android:name="com.android.insecurebankv2.ViewStatement"/>
34    <provider android:authorities="com.android.insecurebankv2.TrackUserContentProvider" android:exported="true"
35      android:name="com.android.insecurebankv2.TrackUserContentProvider"/>
36    <receiver android:exported="true" android:name="com.android.insecurebankv2.MyBroadCastReceiver">
37      <intent-filter>
38        <action android:name="theBroadcast"/>
39      </intent-filter>
40    </receiver>
41    <activity android:exported="true" android:label="@string/title_activity_change_password"
42      android:name="com.android.insecurebankv2.ChangePassword"/>
43    <activity android:configChanges="keyboard|keyboardHidden|orientation|screenLayout|screenSize|smallestScreenSize|uiMode"
44      android:name="com.google.android.gms.ads.AdActivity" android:theme="@android:style/Theme.Translucent"/>
45    <activity android:name="com.google.android.gms.ads.purchase.InAppPurchaseActivity" android:theme="@style/Theme.IAPTheme"/>
46  </application>
47</manifest>
```

Gracias a esto podemos volcar las bases de datos con adb

```
PS F:\platform-tools> .\adb exec-out run-as com.android.insecurebankv2 cat databases/mydb > mydb-copy
PS F:\platform-tools>
```



También podemos ver los archivos almacenados en la aplicación con run-as

```
PS F:\platform-tools> ./adb shell
vbox86p:/ # su shell
:/ $ run-as com.android.insecurebankv2
:/data/user/0/com.android.insecurebankv2 $ ls
cache code_cache databases shared_prefs
```

Esta vulnerabilidad está relacionada con el modo de compilación que ha tenido esta aplicación, al haberse construido en modo ‘debug’ la aplicación resultante está en modo ‘debug’. Para solucionar esto, es necesario volver a compilar la aplicación y construirla con el modo ‘release’ para, de esta forma, eliminar estas características de depuración no deseadas en el proyecto final.

3.2-Insecure Logs

En la clase “DoLogin”, podemos encontrar en el método “postData()”

```
InputStream in = responseBody.getEntity().getContent();
result = convertStreamToString( in );
result = result.replace("\n", "");
if (result != null) {
    if (result.indexOf("Correct Credentials") != -1) {
        Log.d("Successful Login:", "", account=" + username + ":" + password);
        //saveCreds(username, password);
        trackUserLogins();
        Intent pL = new Intent(getApplicationContext(), PostLogin.class);
        pL.putExtra("uname", username);
        startActivity(pL);
    } else {
        Intent xi = new Intent(getApplicationContext(), WrongLogin.class);
        startActivity(xi);
    }
}
```

En este método se crea y guarda un log, sin embargo, se guardan los datos brutos de nombre de usuario y contraseña sin ningún tipo de encriptación por lo que podemos acceder fácilmente a ellos mediante adb logcat

```
05-29 02:52:26.966 397 397 I netd : bandwidthSetInterfaceQuota(radio0, 9223372036854775807) <0.43ms>
05-29 02:52:27.721 10069 10090 D EGL_emulation: app_time_stats: avg=502.82ms min=493.93ms max=511.72ms count=2
05-29 02:52:28.732 10069 10090 D EGL_emulation: app_time_stats: avg=505.10ms min=499.67ms max=510.53ms count=2
05-29 02:52:29.734 10069 10090 D EGL_emulation: app_time_stats: avg=501.17ms min=499.80ms max=502.54ms count=2
05-29 02:52:30.747 10069 10090 D EGL_emulation: app_time_stats: avg=506.41ms min=499.36ms max=513.45ms count=2
05-29 02:52:32.233 10069 10090 D EGL_emulation: app_time_stats: avg=495.47ms min=486.24ms max=500.29ms count=3
05-29 02:52:33.233 10069 10090 D EGL_emulation: app_time_stats: avg=500.07ms min=499.56ms max=500.58ms count=2
05-29 02:52:34.249 10069 10090 D EGL_emulation: app_time_stats: avg=507.85ms min=505.80ms max=509.91ms count=2
05-29 02:52:35.250 10069 10090 D EGL_emulation: app_time_stats: avg=500.25ms min=493.04ms max=507.45ms count=2
05-29 03:52:28.203 0 0 I logd : logdr: UID=0 GID=0 PID=17014 b tail=0 logMask=99 pid=0 start=0nsode
05-29 02:52:36.763 10069 10090 D EGL_emulation: app_time_stats: avg=504.40ms min=488.40ms max=515.15ms count=3
PS F:\platform-tools> ./adb logcat
```

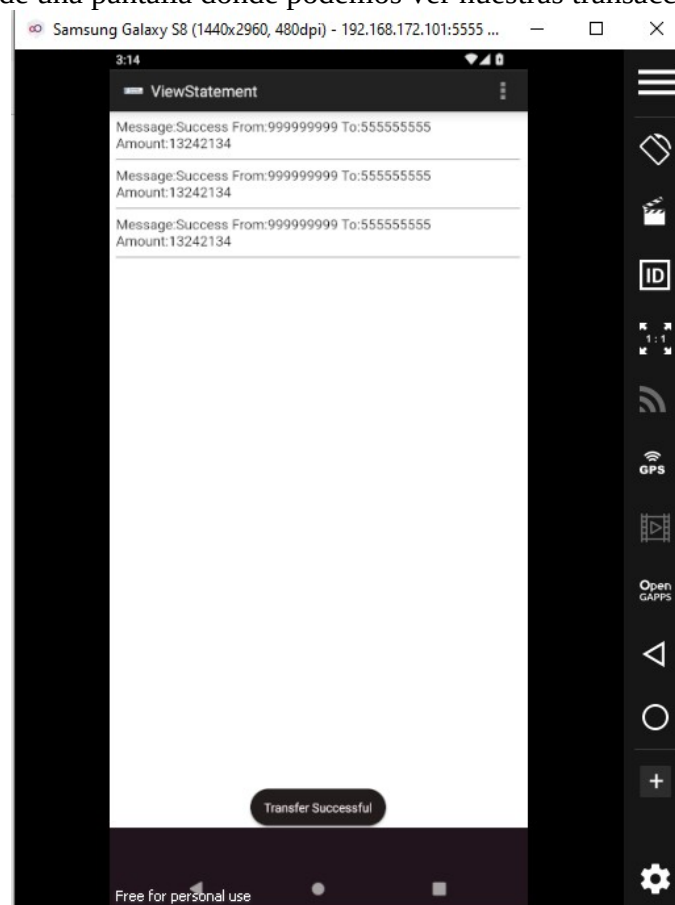
La resolución de esta vulnerabilidad es no guardar la contraseña.

```
InputStream in = responseBody.getEntity().getContent();
result = convertStreamToString( in );
result = result.replace("\n", "");
if (result != null) {
    if (result.indexOf("Correct Credentials") != -1) {
        Log.d("Successful Login:", "", account=" + username);
        //saveCreds(username, password);
        trackUserLogins();
        Intent pL = new Intent(getApplicationContext(), PostLogin.class);
        pL.putExtra("uname", username);
        startActivity(pL);
    } else {
        Intent xi = new Intent(getApplicationContext(), WrongLogin.class);
        startActivity(xi);
    }
}
```

Si se quiere conservar la contraseña en los logs (lo cual sería poco recomendable), lo ideal sería guardar la contraseña ya cifrada y con su salado a ser posible para evitar este tipo de ataques.

3.3-Datos guardados de forma insegura

Esta aplicación consta de una pantalla donde podemos ver nuestras transacciones



Si nos dirigimos al código, podemos encontrar en la clase “ViewStatement”, en el método “onCreate()”, podemos descubrir donde se almacenan estos statements así como la propiedad “setJavaScript” activada.

```
public class ViewStatement extends Activity {
    String uname;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_view_statement);
        Intent intent = getIntent();
        uname = intent.getStringExtra("uname");
        //String statementLocation=Environment.getExternalStorageDirectory()+ "/Statements_" + uname + ".html";
        String FILENAME="Statements " + uname + ".html";
        File fileToCheck = new File(Environment.getExternalStorageDirectory(), FILENAME);
        System.out.println(fileToCheck.toString());
        if (fileToCheck.exists()) {
            //Toast.makeText(this, "Statement Exists!!",Toast.LENGTH_LONG).show();

            WebView mWebView = (WebView) findViewById(R.id.webView1);
            // Location where the statements are stored locally on the device sdcard
            mWebView.loadUrl("file://" + Environment.getExternalStorageDirectory() + "/Statements_" + uname + ".html");
            mWebView.getSettings().setJavaScriptEnabled(true);
            mWebView.getSettings().setSaveFormData(true);
            mWebView.getSettings().setBuiltInZoomControls(true);
            mWebView.setWebViewClient(new MyWebViewClient());
            WebChromeClient cClient = new WebChromeClient();
            mWebView.setWebChromeClient(cClient);
        }
    }
}
```

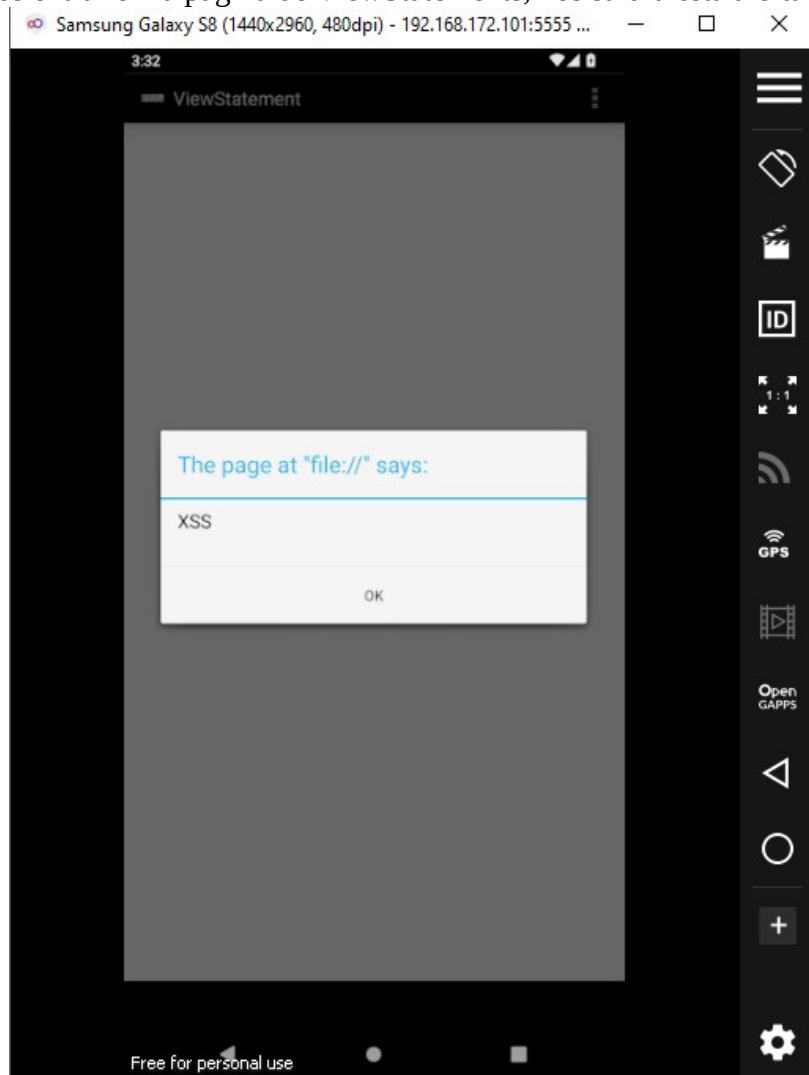
Podemos encontrar este html en “storage/emulated/0”

```
PS F:\platform-tools> ./adb shell
vbox86p:/ # ls
acct      cache      data_mirror  init      mnt        postinstall  sdcard      system      vendor_dlkm
apex      config     debug_ramdisk  init.enviro.rc  odm        proc         second_stage_resources  system_ext
bin       d          dev           linkerconfig  odm_dkkm   product      storage      tmp
bugreports  data      etc           lost+found   oem        sbin         sys          vendor
vbox86p:/ # cd data
data/
vbox86p:/ # cd storage/
vbox86p:/storage # ls
emulated  self
vbox86p:/storage # cd emulated/
vbox86p:/storage/emulated # ls
0  obb
vbox86p:/storage/emulated # cd 0/
vbox86p:/storage/emulated/0 # ls
Alarms  Audiobooks  Documents  Movies  Notifications  Podcasts  Ringtones
Android DCIM      Download  Music   Pictures  Recordings  Statements_jack.html
vbox86p:/storage/emulated/0 #
```

Ahora reemplazaremos este archivo con uno nuevo con XSS inyectado

```
PS F:\platform-tools> ./adb push .\Statements_jack.html /storage/emulated/0
.\Statements_jack.html: 1 file pushed, 0 skipped. 0.0 MB/s (30 bytes in 0.011s)
PS F:\platform-tools> ./adb shell
vbox86p:/ # cd storage/emulated/0/
vbox86p:/storage/emulated/0 # cat Statements_jack.html
vbox86p:/storage/emulated/0 # cat Statements_jack.html
vbox86p:/storage/emulated/0 # cat Statements_jack.html
<script>alert("XSS");</script>vbox86p:/storage/emulated/0 #
```


Si ahora intentamos entrar en la página de ViewStatements, nos saldrá esta alerta de XSS



La mitigación es cambiar a false el parámetro “setJavaScriptEnabled()”

```
public class ViewStatement extends Activity {
    String uname;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_view_statement);
        Intent intent = getIntent();
        uname = intent.getStringExtra("uname");
        //String statementLocation=Environment.getExternalStorageDirectory()+ "/Statements_" + uname + ".html";
        String FILENAME="Statements_" + uname + ".html";
        File fileToCheck = new File(Environment.getExternalStorageDirectory(), FILENAME);
        System.out.println(fileToCheck.toString());
        if (fileToCheck.exists()) {
            //Toast.makeText(this, "Statement Exists!!",Toast.LENGTH_LONG).show();

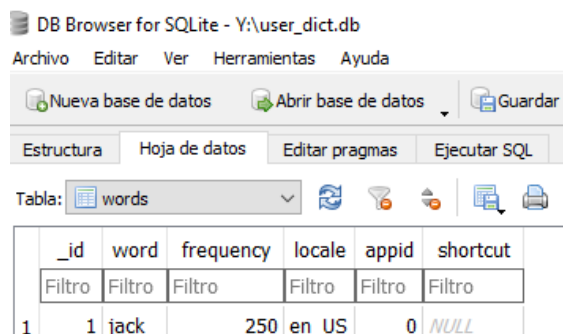
            WebView mWebView = (WebView) findViewById(R.id.webView1);
            // Location where the statements are stored locally on the device sdcard
            mWebView.loadUrl("file://" + Environment.getExternalStorageDirectory() + "/Statements_" + uname + ".html");
            mWebView.getSettings().setJavaScriptEnabled(false);
            mWebView.getSettings().setSaveFormData(true);
            mWebView.getSettings().setBuiltInZoomControls(true);
            mWebView.setWebViewClient(new MyWebViewClient());
            WebChromeClient cClient = new WebChromeClient();
            mWebView.setWebChromeClient(cClient);
        }
    }
}
```

3.4-KeyBoard Cache

Podemos acceder al caché de guardado en la base de datos

```
PS F:\platform-tools> ./adb pull data/data/com.android.providers.userdictionary/databases/user_dict.db
data/data/com.android.providers.userdictionary/databases/user_dict.db: 1 file pulled, 0 skipped, 0.8 MB/s (16384 bytes in 0.020s)
PS F:\platform-tools> ./adb pull data/data/com.android.providers.userdictionary/databases/user_dict.db-journal
data/data/com.android.providers.userdictionary/databases/user_dict.db-journal: 1 file pulled, 0 skipped
PS F:\platform-tools>
```

Podemos acceder a este archivo mediante sqlitebrowser y si se podrían ver los datos guardados en el caché



3.5-Pasteboard Vulnerability

Tras copiar un tecto, este se queda guardado en el clipboard. Con la ayuda de adb, podemos buscar el proceso de 'insecurebank' con adb shell ps y buscar el clipboard del usuario.

```
u0_a80 2335 399 30474888 226864 0 0 S com.android.insecurebankv2
root 2764 2 0 0 worker_thread 0 I [kworker/u8:3-phy0]
u0_a58 2798 400 1025024 93016 0 0 S com.android.webview:webview_service
u0_i9000 2818 1002 1064800 135248 0 0 S com.android.webview:sandboxed_process0:org.chromium.content.app.Sandbo
xedProcessService:0
u0_a58 2852 400 1028644 125224 0 0 S com.android.webview:webview_apk
u0_a55 3042 399 13503588 140860 0 0 S com.android.messaging
root 3261 2 0 0 worker_thread 0 I [kworker/u8:0-events_unbound]
root 3427 2 0 0 worker_thread 0 I [kworker/u8:1-phy0]
root 3555 492 10802892 3492 0 0 R ps
PS F:\platform-tools> ./adb shell su u0_a80 service call clipboard 2 s16 com.android.insecurebankv2
Result: Parcel(
  0x00000000: ffffffff 00000073 00650052 00750071 '...s...R.e.q.u.'
  0x00000010: 00720069 00730065 00530020 00540045 'i.r.e.s...S.E.T.'
  0x00000020: 0043005f 0049004e 005f0050 004f0053 '...C.L.I.P...S.O.'
  0x00000030: 00520055 00450045 00700020 00720065 'U.R.C.E...p.e.r.'
  0x00000040: 0069006d 00730073 006f0069 003a006e 'm.i.s.s.i.o.n...S.'
  0x00000050: 004e0020 00690065 00680074 00720065 '...N.e.i.t.h.e.r.'
  0x00000060: 00750020 00650073 00200072 00300031 '...u.s.e.r...l.O.'
  0x00000070: 00380030 00200030 006f006e 00200072 '0.8.0...n.o.r...'
  0x00000080: 00750063 00720072 006e0065 00200074 'c.u.r.r.e.n.t...'
  0x00000090: 00720070 0063006f 00730065 00200073 'p.r.o.c.e.s.s...'
  0x000000a0: 00610068 00200073 006e0061 00720064 'h.a.s...a.n.d.r.'
  0x000000b0: 0069006f 002e0064 00650070 006d0072 'o.i.d...p.e.r.m.'
  0x000000c0: 00730069 00690073 006e006f 0053002e 'i.s.s.i.o.n...S.'
  0x000000d0: 00540045 0043005f 0049004c 005f0050 'E.T...C.L.I.P...'
  0x000000e0: 004f0053 00520055 00450043 0000002e 'S.O.U.R.C.E....'
  0x000000f0: 00000360 000001aa 00610009 00200074 '.....a.t...'
  0x00000100: 006e0061 00720064 0069006f 002e0064 'a.n.d.r.o.i.d...'
  0x00000110: 00700061 002e0070 006f0043 0074006e 'a.p.p...C.o.n.t.'
  0x00000120: 00780065 00490074 0070006d 002e006c 'e.x.t.I.m.p.l...'
)
```

En este caso no he encontrado la cuenta copiada en el clipboard, pero pueden ser filtrados contenidos sensibles de esta forma.

Para mitigar este error, podemos crear un método “clearClipboard()” que limpie el clipboard y llamarlo cuando consideremos conveniente

```
import android.content.ClipboardManager;

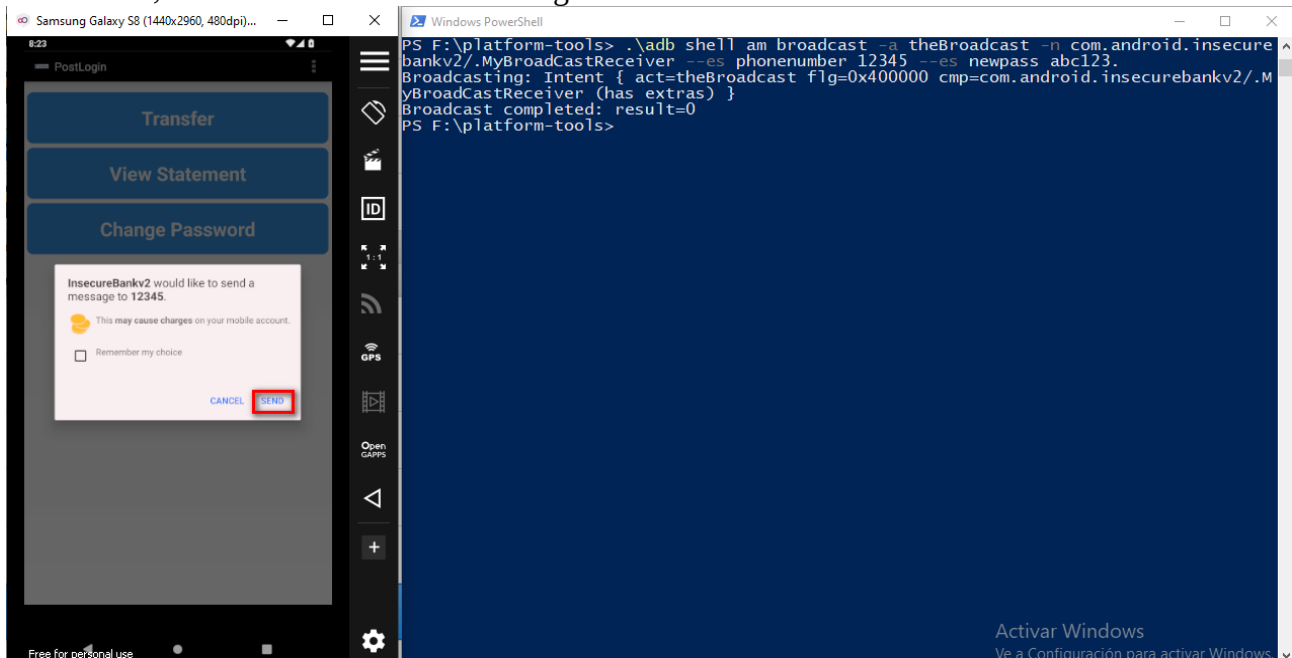
private void clearClipboard() {
    if (clipboardManager != null && clipboardManager.hasPrimaryClip()) {
        clipboardManager.setPrimaryClip(ClipData.newPlainText("", ""));
    }
}
```

3.6-Exported Broadcast Receiver

En el AndroidManifest, podemos encontrar “MyBroadCastReceiver” con “exported:true”, por lo que podemos llamarlo con adb

```
<provider android:name=".TrackUserContentProvider" android:exported="true" android:authorities="com.android.insecurebankv2.TrackUserContentProvider">
</provider>
<receiver android:name=".MyBroadCastReceiver" android:exported="true">
  <intent-filter>
```

Al llamarlo, recibiremos una ventana emergente.



La mitigación es cambiar la propiedad ‘exported’ a false

```
<receiver
  android:name=".MyBroadCastReceiver"
  android:exported="false" >
  <intent-filter>
    <action android:name="theBroadcast" >
    </action>
  </intent-filter>
</receiver>
```


3.7-Exported TrackUsersProvider

Podemos encontrar “TrackUserContentProvider” exportada.

```
<activity android:name=".DoLogin" android:label="@string/title_activity_do_login"> </activity>
<activity android:name=".PostLogin" android:label="@string/title_activity_post_login" android:exported="false"> </activity>
<activity android:name=".WrongLogin" android:label="@string/title_activity_wrong_login"> </activity>
<activity android:name=".DoTransfer" android:label="@string/title_activity_do_transfer" android:exported="false"> </activity>
<activity android:name=".ViewStatement" android:label="@string/title_activity_view_statement" android:exported="false"> </activity>
<provider android:name=".TrackUserContentProvider" android:exported="true" android:authorities="com.android.insecurebankv2.TrackUserContentProvider">
</provider>
```

Podemos aprovechar esto para obtener un registro de inicio de sesión de los usuarios guardado en la siguiente dirección:

```
public class TrackUserContentProvider extends ContentProvider {

    // This content provider vuln is a modified code from www.androidpentesting.com

    static final String PROVIDER_NAME = "com.android.insecurebankv2.TrackUserContentProvider";
    // The Content provider that handles all the tracked user history
    static final String URL = "content://" + PROVIDER_NAME + "/trackerusers";
    static final Uri CONTENT_URI = Uri.parse(URL);
    static final String name = "name";
    static final int uriCode = 1;
    static final UriMatcher uriMatcher;
    private static HashMap < String, String > values;
    private SQLiteDatabase db;
    static final String DATABASE_NAME = "mydb";
    static final String TABLE_NAME = "names";
    static final int DATABASE_VERSION = 1;
    static final String CREATE_DB_TABLE = "CREATE TABLE " + TABLE_NAME + " (id INTEGER PRIMARY KEY AUTOINCREMENT, " + " name TEXT NOT NULL);";
```

```
PS F:\platform-tools> ./adb shell content query --uri content://com.android.insecurebankv2.TrackUserContentProvider/trackerusers
Row: 0 id=3, name=devadmin
Row: 1 id=4, name=devadmin
Row: 2 id=5, name=devadmin
Row: 3 id=2, name=dinesh
Row: 4 id=1, name=jack
Row: 5 id=6, name=jack
Row: 6 id=7, name=jack
Row: 7 id=8, name=jack
Row: 8 id=9, name=jack
Row: 9 id=10, name=jack
Row: 10 id=11, name=jack
Row: 11 id=12, name=jack
Row: 12 id=13, name=jack
Row: 13 id=14, name=jack
Row: 14 id=15, name=jack
```

Podemos reparar esta vulnerabilidad cambiando su respectivo campo a ‘false’

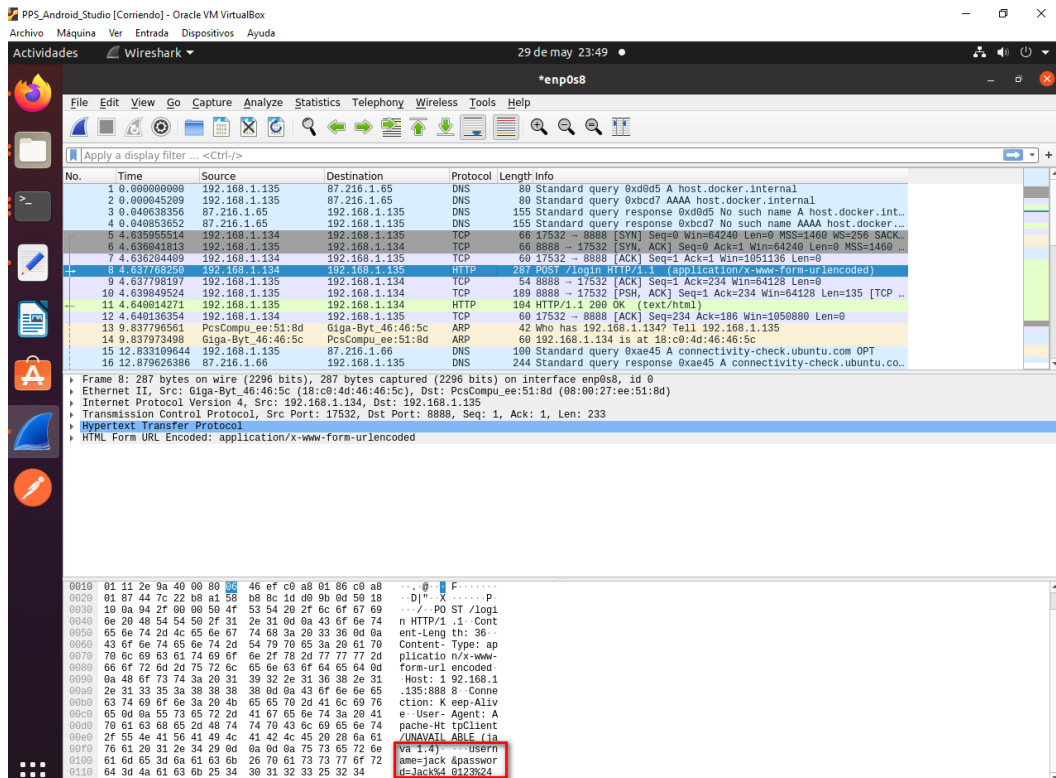
```
<provider
    android:name=".TrackUserContentProvider"
    android:authorities="com.android.insecurebankv2.TrackUserContentProvider"
    android:exported="false" >
</provider>
```

3.8-Utilización de protocolo de red no seguro (HTTP)

En las clases “DoLogin”, “DoTransfer” y “ChangePassword”, podemos encontrar que el protocolo utilizado es http

```
public class DoLogin extends Activity {
    String responseString = null;
    // Stores the username passed by the calling intent
    String username;
    // Stores the password passed by the calling intent
    String password;
    String result;
    String superSecurePassword;
    String rememberme_username, rememberme_password;
    public static final String MYPREFS = "mySharedPreferences";
    String serverip = "";
    String serverport = "";
    String protocol = "http://";
    BufferedReader reader;
    SharedPreferences serverDetails;
```

Podemos comprobar esto si abrimos wireshark para capturar el tráfico generado por nuestras acciones.



Como podemos ver en la captura anterior, se han filtrado el nombre de usuario y contraseña debido al uso de http.

El remedio es reemplazar el protocolo http por su contraparte segura (https) en todas aquellas clases que la utilicen (DoLogin, DoTransfer y ChangePassword).

```

public class Dologin extends Activity {
    String responseString = null;
    // Stores the username passed by the calling intent
    String username;
    // Stores the password passed by the calling intent
    String password;
    String result;
    String superSecurePassword;
    String rememberme_username, rememberme_password;
    public static final String MYPREFS = "mySharedPreferences";
    String serverip = "";
    String serverport = "";
    String protocol = "https://";
    BufferedReader reader;
    SharedPreferences serverDetails;
}

```

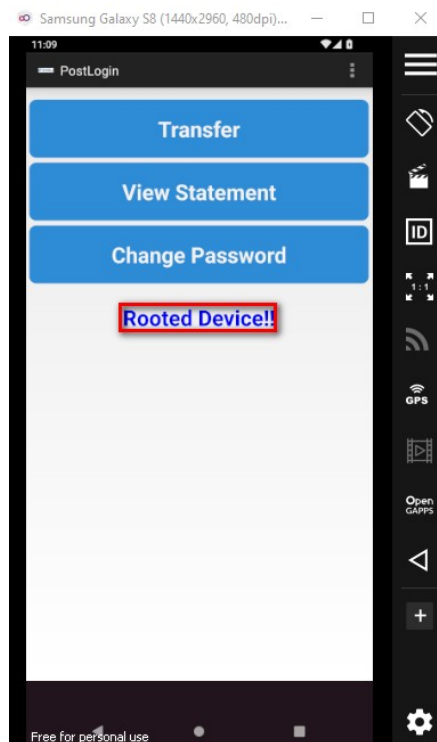
3.9-Vulnerable a ataques de fuerza bruta/diccionario

Al no existir ninguna medida que ralentice el inicio de sesión, podríamos utilizar programas como burpsuite para realizar ataques de diccionario o de fuerza bruta para descifrar las credenciales.

Esto se mitiga parcialmente cambiando el protocolo http por su contraparte segura, pero sería necesario introducir algún tipo de ralentización a la hora de inicio de sesión para evitar que se realice de forma automática.

3.10-RootDetection Bypass

Al iniciar sesión con un usuario, tendremos un mensaje diciendo que nuestro dispositivo está rooteado.

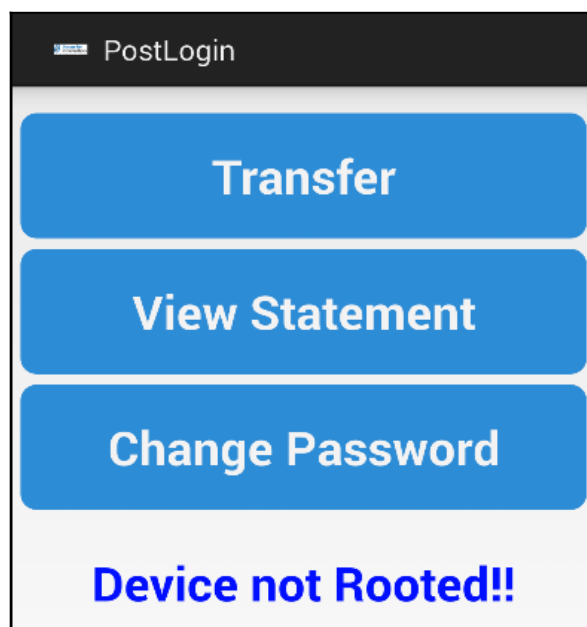


Esta comprobación la podemos encontrar en la clase “PostLogin” en el método “showRootStatus”

```
void showRootStatus() {  
    boolean isrooted = doesSuperuserApkExist("/system/app/Superuser.apk") ||  
        doesSUexist();  
    if(isrooted==true)  
    {  
        root_status.setText("Rooted Device!!");  
    }  
    else  
    {  
        root_status.setText("Device not Rooted!!");  
    }  
}
```

Podemos utilizar la herramienta ‘frida’ para cambiar el valor de estos booleanos a false

```
console.log("Script loaded successfully "); // console.log is used print messages to the console  
Java.perform(function x() {  
    console.log("Inside java perform function");  
    // Class the function belongs to.  
    var my_class = Java.use("com.android.insecurebankv2.PostLogin");  
  
    my_class.doesSUexist.implementation = function (x) { //hooking the function  
        console.log("*****SU Root Check Bypassed*****");  
        return false  
    };  
  
    my_class.doesSuperuserApkExist.implementation = function (x) { //hooking the function  
        console.log("*****SuperuserAPK Root Check Bypassed*****");  
        return false  
    };  
});
```



Una posible solución a este problema puede ser implementar técnicas de ofuscación en el código para cambiar el nombre de estos métodos y dificultar este proceso (pero no es totalmente efectivo).