

ANALISIS DISPOSITIVOS MOVILES



BRUNO LÓPEZ BARCIA | ANALIZADOR | 28/05/2023

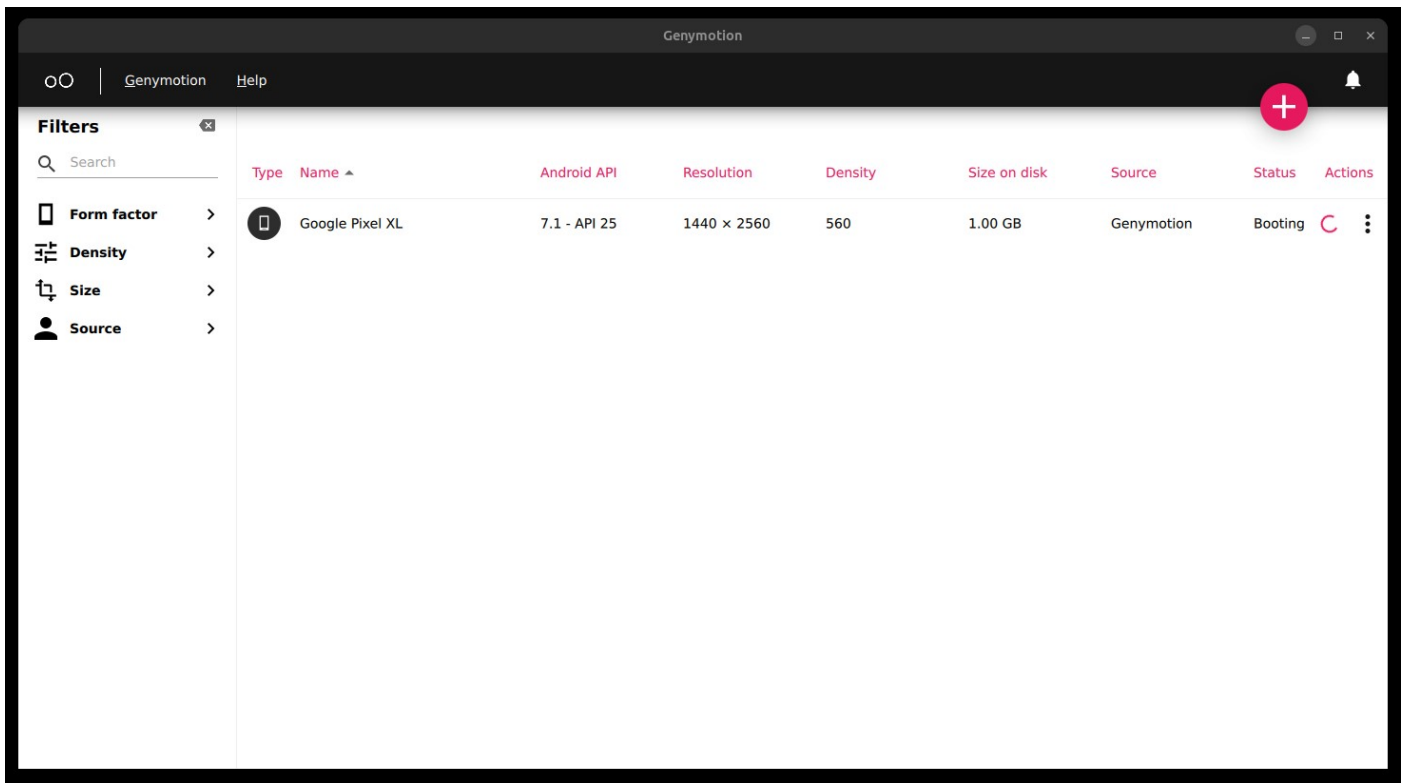
CONTENIDO

PREPARACIÓN DEL ENTORNO	1
ANALISIS ESTATICO.....	2
ANALISIS DINAMICO	8
BYPASSING LOGIN PAGE.....	8
LOGIN ADMIN	9
CREATE USER BUTTON	10
CONTENT PROVIDER.....	11
BROADCAST RECIVER	11
LOGCAT DE LA APLICACION	12
UNSECURE PASSWORD STORAGE.....	13
ACCESS TO DB	14
UNSECURE DATA STORAGE.....	15
INSECURE CONNECTIONS	16
CORRECCION VULNERABILIDADES	17
BYPASSING LOGIN PAGE.....	17
LOGIN ADMIN	18
CREATE USER BUTTON	19
BROADCAST RECEIVER.....	20
LOGCAT DE LA aplicación.....	21
UNSECURE PASSWORD STORAGE.....	21
ACCESS TO DB	21
CONTENT PROVIDER.....	22
UNSECURE DATA STORAGE.....	22
INSECURE CONNECTIONS	23
EXECUTION JS ON VIEW STATEMENTS.....	23
REPOSITORIO GITHUB	24

PREPARACIÓN DEL ENTORNO

Lo primero que debemos de hacer es preparar el entorno; para ello debemos de iniciar el MobSF y una máquina virtual en GenyMotion.

```
brunolb@UbuntuBruno: ~/Imágenes/Capturas de pantalla
brunolb@UbuntuBruno: ~/Imágenes/Capturas de pantalla$ mobsf
[2023-05-28 16:26:47 +0200] [7278] [INFO] Starting gunicorn 20.1.0
[2023-05-28 16:26:47 +0200] [7278] [INFO] Listening at: http://127.0.0.1:8080 (7278)
[2023-05-28 16:26:47 +0200] [7278] [INFO] Using worker: gthread
[2023-05-28 16:26:47 +0200] [7325] [INFO] Booting worker with pid: 7325
```



The screenshot shows the Genymotion application window. On the left, there is a 'Filters' sidebar with options for 'Form factor', 'Density', 'Size', and 'Source'. The main area displays a table of virtual devices. The table has columns for Type, Name, Android API, Resolution, Density, Size on disk, Source, Status, and Actions. One device is listed: 'Google Pixel XL' with API level 25, resolution 1440 x 2560, density 560, size 1.00 GB, and status 'Booting'.

Type	Name	Android API	Resolution	Density	Size on disk	Source	Status	Actions
📱	Google Pixel XL	7.1 - API 25	1440 x 2560	560	1.00 GB	Genymotion	Booting	⋮

ANALISIS ESTATICO

En el análisis estático encontramos las siguientes cosas.

The screenshot shows the MobSF Static Analysis tool interface. The left sidebar contains navigation options: Information, Scan Options, Signer Certificate, Permissions, Android API, Browsable Activities, Security Analysis, Malware Analysis, Reconnaissance, Components, PDF Report, Print Report, and Dynamic Analysis Report. The main content area displays the following information:

- APP SCORES:** Security Score 33/100, Trackers Detection 3/428, MobSF Scorecard.
- FILE INFORMATION:** File Name InsecureBankv2.apk, Size 3.3MB, MD5 5ee4829065640f9c936ac861d1650ffc, SHA1 80b53f80a3c9e6bdf98311f5b26ccddcd1bf0a98, SHA256 b18af2a0e44d7634bbcd93664d9c78a2695e050393fcb5e8b91f902d194a4.
- APP INFORMATION:** App Name InsecureBankv2, Package Name com.android.insecurebankv2, Main Activity com.android.insecurebankv2.LoginActivity, Target SDK 22, Min SDK 15, Max SDK, Android Version Name 1.0, Android Version Code 1.
- ACTIVITIES:** 10 activities, 4 exported.
- SERVICES:** 0 services, 0 exported.
- RECEIVERS:** 2 receivers, 1 exported.
- PROVIDERS:** 1 provider, 1 exported.
- SCAN OPTIONS:** Rescan, Manage Suppressions, Start Dynamic Analysis.
- DECOMPILED CODE:** View AndroidManifest.xml, View Source, View Smali, Download Java Code, Download Smali Code, Download APK.
- SIGNER CERTIFICATE:** (Section header visible).

Como se puede ver en la captura anterior hay 10 actividades en la aplicación de las cuales 4 están con la opción exported a true. También nos encuentra dos receivers uno de los cuales tiene la opción exported a true. Por último, nos encuentra que la aplicación tiene un Provider el cual tiene le exported a true.

Si miramos las Activitys vemos lo siguiente:

The screenshot shows the MobSF Static Analysis tool interface with the 'ACTIVITIES' section selected. The list of activities is as follows:

- com.android.insecurebankv2.LoginActivity
- com.android.insecurebankv2.FilePreferActivity
- com.android.insecurebankv2.DoLogin
- com.android.insecurebankv2.PostLogin
- com.android.insecurebankv2.WrongLogin
- com.android.insecurebankv2.DoTransfer
- com.android.insecurebankv2.ViewStatement
- com.android.insecurebankv2.ChangePassword
- com.google.android.gms.ads.AdActivity
- com.google.android.gms.ads.purchase.InAppPurchaseActivity

Estas son las activitys; las que tienen el exported a true las utilizaremos en el análisis dinámico.

En cuanto a los receivers y los provider encontramos lo siguiente:

The screenshot shows the MobSF Static Analysis tool interface with the 'RECEIVERS' and 'PROVIDERS' sections selected. The list of receivers and providers is as follows:

- RECEIVERS:** com.android.insecurebankv2.MyBroadcastReceiver, com.google.android.gms.wallet.EnableWalletOptimizationReceiver.
- PROVIDERS:** com.android.insecurebankv2.TrackUserContentProvider.

En cuanto al análisis del manifest de la aplicación encontramos lo siguiente:

Static Analysis

localhost:8000/static_analyzer/?name=insecureBankv2.apk&checksum=5ee4829065640f9c936ac861d1650ffc&type=apk#activities

MobSF

RECENT SCANSSTATIC ANALYZERDYNAMIC ANALYZERREST APIDONATE▼DOCSABOUTSearch MD5

Static Analyzer

Information

Scan Options

Signer Certificate

Permissions

Android API

Browsable Activities

Security Analysis

Malware Analysis

Reconnaissance

Components

PDF Report

Print Report

Dynamic Analysis Report

CERTIFICATE ANALYSIS

HIGH1WARNING0INFO1

Search:

TITLE	SEVERITY	DESCRIPTION
Application vulnerable to Janus Vulnerability	high	Application is signed with v1 signature scheme, making it vulnerable to Janus vulnerability on Android 5.0-8.0, if signed only with v1 signature scheme. Applications running on Android 5.0-7.0 signed with v1, and v2/v3 scheme is also vulnerable.
Signed Application	info	Application is signed with a code signing certificate

Showing 1 to 2 of 2 entries

Previous1Next

MANIFEST ANALYSIS

HIGH1WARNING2INFO0SUPPRESSED0

Search:

NO	ISSUE	SEVERITY	DESCRIPTION	OPTIONS
1	App can be installed on a vulnerable Android version [minSdk=15]	warning	This application can be installed on an older version of android that has multiple unfixed vulnerabilities. Support an Android version > 8, API 26 to receive reasonable security updates.	
2	Debug Enabled For App [android:debuggable=true]	high	Debugging was enabled on the app which makes it easier for reverse engineers to hook a debugger to it. This allows dumping a stack trace and accessing debugging helper classes.	
3	Application Data can be Backed up [android:allowBackup=true]	warning	This flag allows anyone to backup your application data via adb. It allows users who have enabled USB debugging to copy application data off of the device.	

Showing 1 to 3 of 3 entries

Previous1Next

Como vemos nos avisa que en el archivo AndroidManifest.xml, esta tanto la opción debugged como la opción allowBackup en true, lo que podría permitir al atacante obtener información valiosa.

En cuanto los permisos de la aplicación vemos lo siguiente:

Static Analysis

localhost:8000/static_analyzer/?name=insecureBankv2.apk&checksum=5ee4829065640f9c936ac861d1650ffc&type=apk#activities

MobSF

RECENT SCANSSTATIC ANALYZERDYNAMIC ANALYZERREST APIDONATE▼DOCSABOUTSearch MD5

Static Analyzer

Information

Scan Options

Signer Certificate

Permissions

Android API

Browsable Activities

Security Analysis

Malware Analysis

Reconnaissance

Components

PDF Report

Print Report

Dynamic Analysis Report

APPLICATION PERMISSIONS

Search:

PERMISSION	STATUS	INFO	DESCRIPTION
android.permission.ACCESS_COARSE_LOCATION	dangerous	coarse (network-based) location	Access coarse location sources, such as the mobile network database, to determine an approximate phone location, where available. Malicious applications can use this to determine approximately where you are.
android.permission.ACCESS_NETWORK_STATE	normal	view network status	Allows an application to view the status of all networks.
android.permission.GET_ACCOUNTS	dangerous	list accounts	Allows access to the list of accounts in the Accounts Service.
android.permission.INTERNET	normal	full Internet access	Allows an application to create network sockets.
android.permission.READ_CONTACTS	dangerous	read contact data	Allows an application to read all of the contact (address) data stored on your phone. Malicious applications can use this to send your data to other people.
android.permission.READ_PROFILE	dangerous	read the user's personal profile data	Allows an application to read the user's personal profile data.
android.permission.SEND_SMS	dangerous	send SMS messages	Allows application to send SMS messages. Malicious applications may cost you money by sending messages without your confirmation.
android.permission.USE_CREDENTIALS	dangerous	use the authentication credentials of an account	Allows an application to request authentication tokens.
android.permission.WRITE_EXTERNAL_STORAGE	dangerous	read/modify/delete external storage contents	Allows an application to write to external storage.

Showing 1 to 9 of 9 entries

Previous1Next

ANDROID API

Search:

Analizando el Manifest.xml vemos lo siguiente:

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <manifest android:versionCode="1" android:versionName="1.0" package="com.android.insecurebankv2" platformBuildVersionCode="22" platformBuildVersionName="5.1.1-1819727"
3. xmlns:android="http://schemas.android.com/apk/res/android">
4.     <uses-sdk android:minSdkVersion="15" android:targetSdkVersion="22" />
5.     <uses-permission android:name="android.permission.INTERNET" />
6.     <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
7.     <uses-permission android:name="android.permission.SEND_SMS" />
8.     <uses-permission android:name="android.permission.USE_CREDENTIALS" />
9.     <uses-permission android:name="android.permission.GET_ACCOUNTS" />
10.    <uses-permission android:name="android.permission.READ_PROFILE" />
11.    <uses-permission android:name="android.permission.READ_CONTACTS" />
12.    <android:uses-permission android:name="android.permission.READ_PHONE_STATE" />
13.    <android:uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" android:maxSdkVersion="18" />
14.    <android:uses-permission android:name="android.permission.READ_CALL_LOG" />
15.    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
16.    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
17.    <uses-feature android:glEsVersion="0x00200000" android:required="true" />
18.    <application android:theme="@android:style/Theme.Holo.Light.DarkActionBar" android:label="@string/app_name" android:icon="@mipmap/ic_launcher" android:debuggable="true" android:allowBackup="true">
19.        <activity android:label="@string/app_name" android:name="com.android.insecurebankv2.LoginActivity">
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:windowSoftInputMode="adjustNothing|stateVisible" />
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.TrackUserContentProvider" />
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|keyboardHidden|orientation|screenLayout|screenSize|smallestScreenSize|uiMode" />
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>
```

Como podemos ver las hay algunas activitys que están con el valor exported a True lo que significa que pueden ser llamadas por otras funciones, aunque sean ajenas a la App. Como veremos luego seremos capaces de saltarnos el login gracias a este método.

En cuanto al receiver y el provider tendrían el mismo problema.

También podemos ver que tanto el debugmode como el backup están activados lo cual nos permitiría acceder a archivos y logs que no se debería de poder acceder.

En el archivo llamado LoginActivity podemos ver el siguiente código:

```
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);
login_buttons.setOnClickListener(new View.OnClickListener() {
```

```
1 usage
protected void createUser() {
    Toast.makeText(this, "Create User functionality is still Work-In-Progress!!", 1).show();
}
```

Esto lo que hace es mostrar un botón de sirve para crear un usuario en caso de que tenga un valor en yes en un string. El botón por ahora como podemos ver en la segunda captura no hace nada.

Aquí podemos ver que hay un método que nos permite guardar las credenciales. Esto puede ser malo; ya que se guarda en un archivo y es fácilmente crackeable.

```
1 usage
private void saveCreds(String username, String password) throws UnsupportedOperationException, InvalidKeyException, No
    // 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();
}
```

Lo siguiente que hemos encontrado es, un if que en caso de que el nombre de usuario sea “devadmin” no necesitas contraseña para hacer login.

```
nameValuePairs.add(new BasicNameValuePair("username", username));
nameValuePairs.add(new BasicNameValuePair("password", password));
HttpResponse responseBody;
if (username.equals("devadmin")) {
    httpPost2.setEntity(new UrlEncodedFormEntity(nameValuePairs));
    // Execute HTTP Post Request
    responseBody = httpClient.execute(httpPost2);
} else {
    httpPost.setEntity(new UrlEncodedFormEntity(nameValuePairs));
    // Execute HTTP Post Request
    responseBody = httpClient.execute(httpPost);
}
```

Un poco más arriba se encuentra un comentario donde aparece tanto un usuario como una contraseña.

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

```
1 usage
private void saveCreds(String username, String password) throws UnsupportedOperationException, InvalidKeyException, No
// 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();
}
```

Como podemos ver en la captura anterior el cifrado de la contraseña y del usuario es insuficiente debido a que utilizan, AES y base64 respectivamente. Como podemos ver esto se guarda en MYPREFS, que es un archivo al cual se puede acceder como veremos posteriormente.

Otra vulnerabilidad que hemos encontrado es la utilización del protocolo http para comunicarse entre cliente y servidor. Esto podemos encontrarlo en los archivos DoLogin, ChangePassword, DoTransfer.

```
String protocol = "http://";
```

Este String debería de ser https en lugar de http.

También hemos encontrado que almacena información que no es necesaria en el log; esto es debido a que almacena tanto el usuario como la contraseña de la persona que hace login.

```
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);
    }
}
```

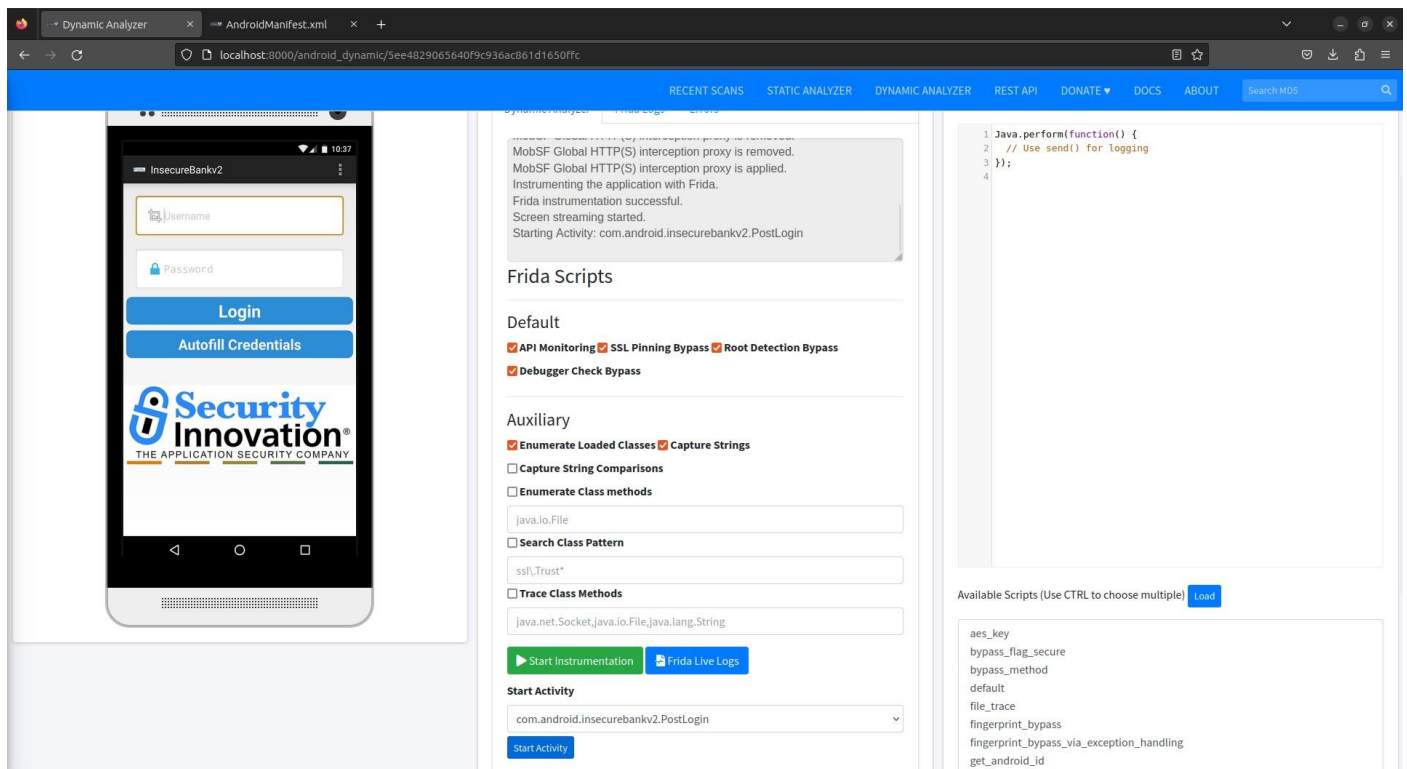
También hemos encontrado que en el ViewState se permite modificar el amount de la transferencia del archivo y añadir otra cosa, incluso javascript que lo va a ejecutar.

```
WebView mWebView = (WebView) findViewById(R.id.webView1);
// Location where the statements are stored locally on the device sdcard
mWebView.loadUrl("file://" + Environment.getExternalStorageDirectory() + "/Statements_" +
mWebView.getSettings().setJavaScriptEnabled(true);
mWebView.getSettings().setSaveFormData(true);
mWebView.getSettings().setBuiltInZoomControls(true);
mWebView.setWebViewClient(new MyWebViewClient());
WebChromeClient cClient = new WebChromeClient();
mWebView.setWebChromeClient(cClient);
} else
{
    Intent gobacktoPostLogin =new Intent(this,PostLogin.class);
    startActivity(gobacktoPostLogin);
}
```

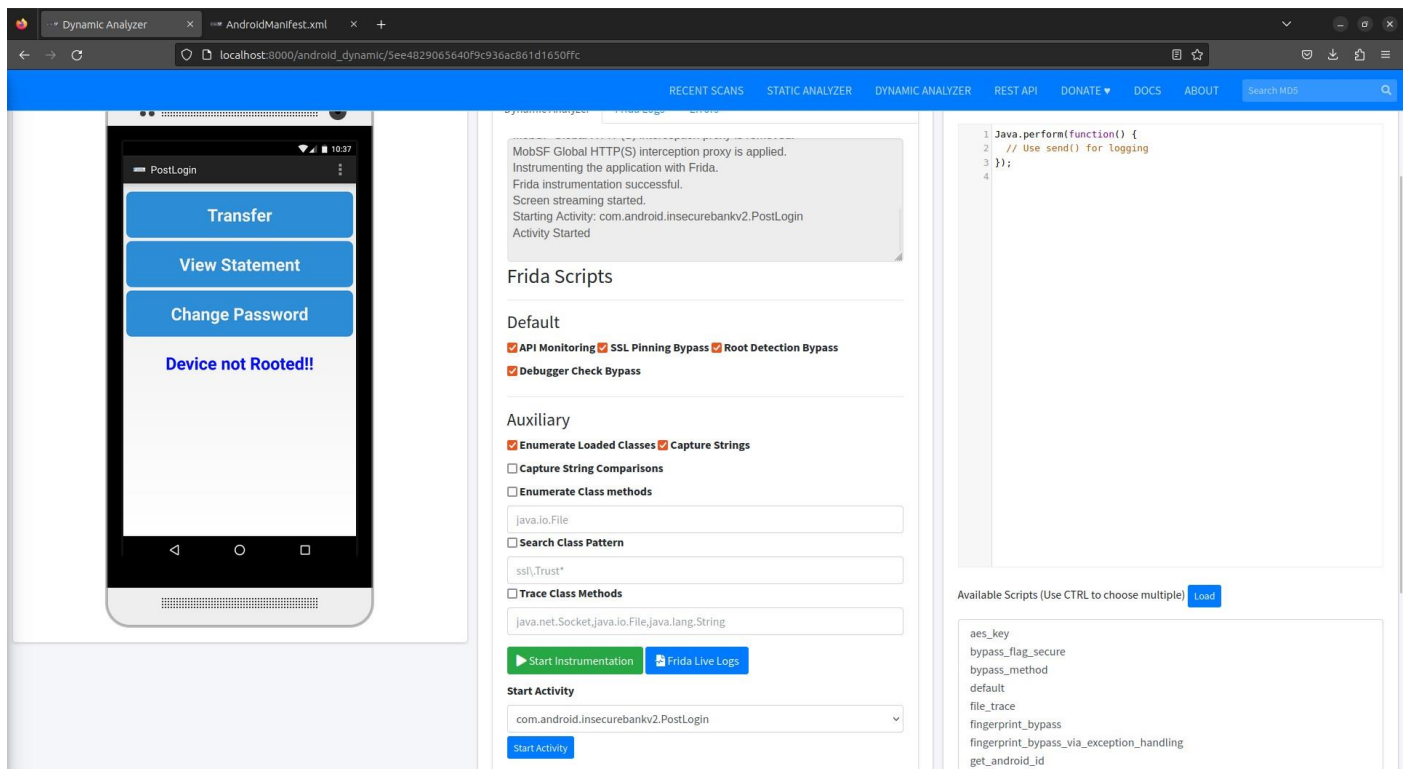
ANALISIS DINAMICO

BYPASSING LOGIN PAGE

Para realizar esto nos ayudaremos del mobsf; el cual hemos conectado a un Android virtualizado en Genymotion. El bypassing del login se hace llamando al método PostLogin, el cual vimos anteriormente que tenía el exported a True, por lo que puede ser llamado por cualquier método.



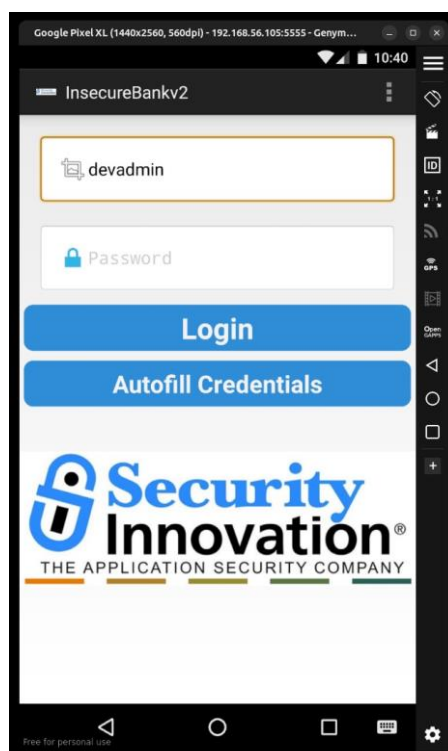
Como se puede ver en la opción Start Activity tenemos seleccionado el método PostLogin. Si le damos al botón veremos que podemos hacer login sin necesidad de poner un usuario y contraseña.



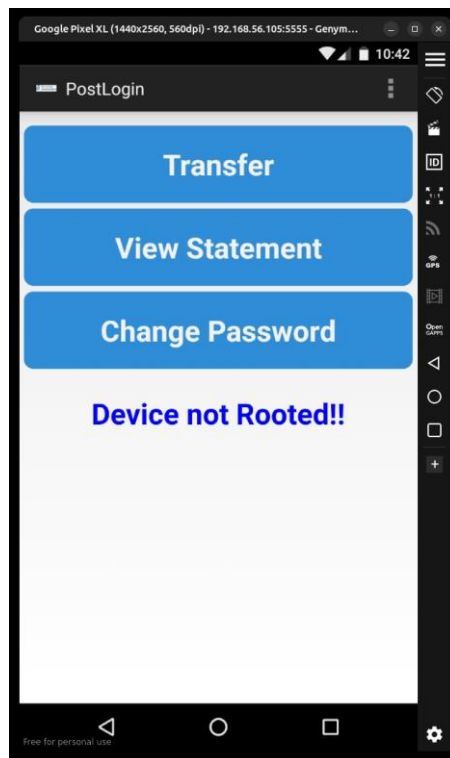
Aquí hemos realizado la activity y cómo podemos ver hemos pasado el login correctamente, sin necesidad de contraseña.

LOGIN ADMIN

Como pudimos ver anteriormente en el análisis estático, había un trozo de función donde gracias a un if podíamos iniciar sesión sin necesidad de contraseña con el usuario “devadmin”



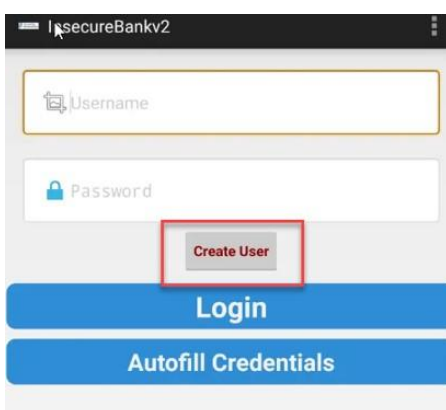
Si ahora probamos a darle a login sin poner contraseña veremos que podemos iniciar sesión correctamente.



CREATE USER BUTTON

Como vimos anteriormente en el análisis estático había un método que mostraba un botón de crear un usuario en caso de que hubiera un string en “yes”. Si modificamos el valor del string a “yes” y volvemos a empaquetar la aplicación y la ejecutamos veremos que a nosotros también nos muestra el botón.

```
<string name="create_calendar_message">Allow Ad to create a calendar event</string>
<string name="create_calendar_title">Create calendar event</string>
<string name="decline">Decline</string>
<string name="hello_world">Hello world!</string>
<string name="is_admin">yes</string>
<string name="loginscreen_password">Password:</string>
<string name="loginscreen_username">Username:</string>
<string name="pref_submit">Submit:</string>
<string name="server_ip">Server IP:</string>
```



CONTENT PROVIDER

Este content provider nos permitirá obtener una lista de usuarios que hayan iniciado sesión en la aplicación; podemos llamar al broadcast desde ADB. Debemos de ejecutar el siguiente comando.

```
Símbolo del sistema x + v
C:\Users\bruno\Downloads\platform-tools_r34.0.3-windows\platform-tools>adb shell content query --uri content://com.android.insecurebankv2.TrackUserContentProvider/trackerusers
Row: 0 id=1, name=jack
```

Como podemos ver nos saca un listado de todos los usuarios que han iniciado sesión en la aplicación. Lo que hace el comando anterior es ejecutar el content provider de la aplicación el cual tiene el valor en exported=True en el AndroidManifest.

BROADCAST RECIVER

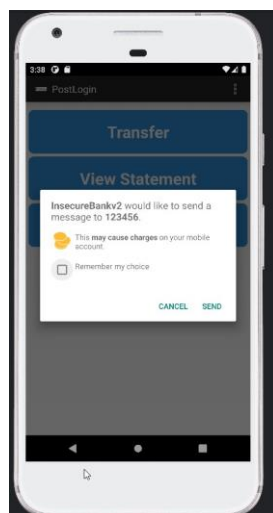
El broadcast receiver lo que permite es cambiar la contraseña del usuario que esta logueado y le envía un mensaje con la nueva password.

Como en el caso anterior lo que debemos de hacer es llamar al método que lo realiza; esto lo haremos mediante ADB.

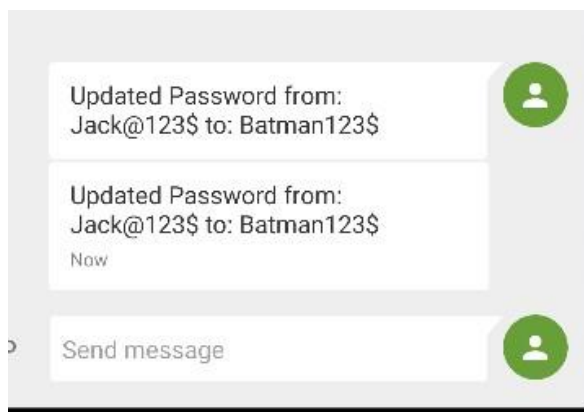
```
Command Prompt for vcl
C:\Users\bruno\Downloads\platform-tools_r34.0.3-windows\platform-tools>.adb.exe shell am broadcast -a theBroadcast -n com.android.insecurebankv2/.MyBroadcastReceiver --es phonenum 123456 --es newpass Batman123$
Broadcasting: Intent { act=theBroadcast flg=0x400000 cmp=com.android.insecurebankv2/.MyBroadcastReceiver (has extras) }
Broadcast completed: result=0
C:\Users\bruno\Downloads\platform-tools_r34.0.3-windows\platform-tools>
```

Este comando es el que llama al Broadcast receiver y le dice que cambie la contraseña a Batman123\$.

Si ahora miramos la aplicación nos aparece un mensaje de si queremos aceptar él envió de mensajes.



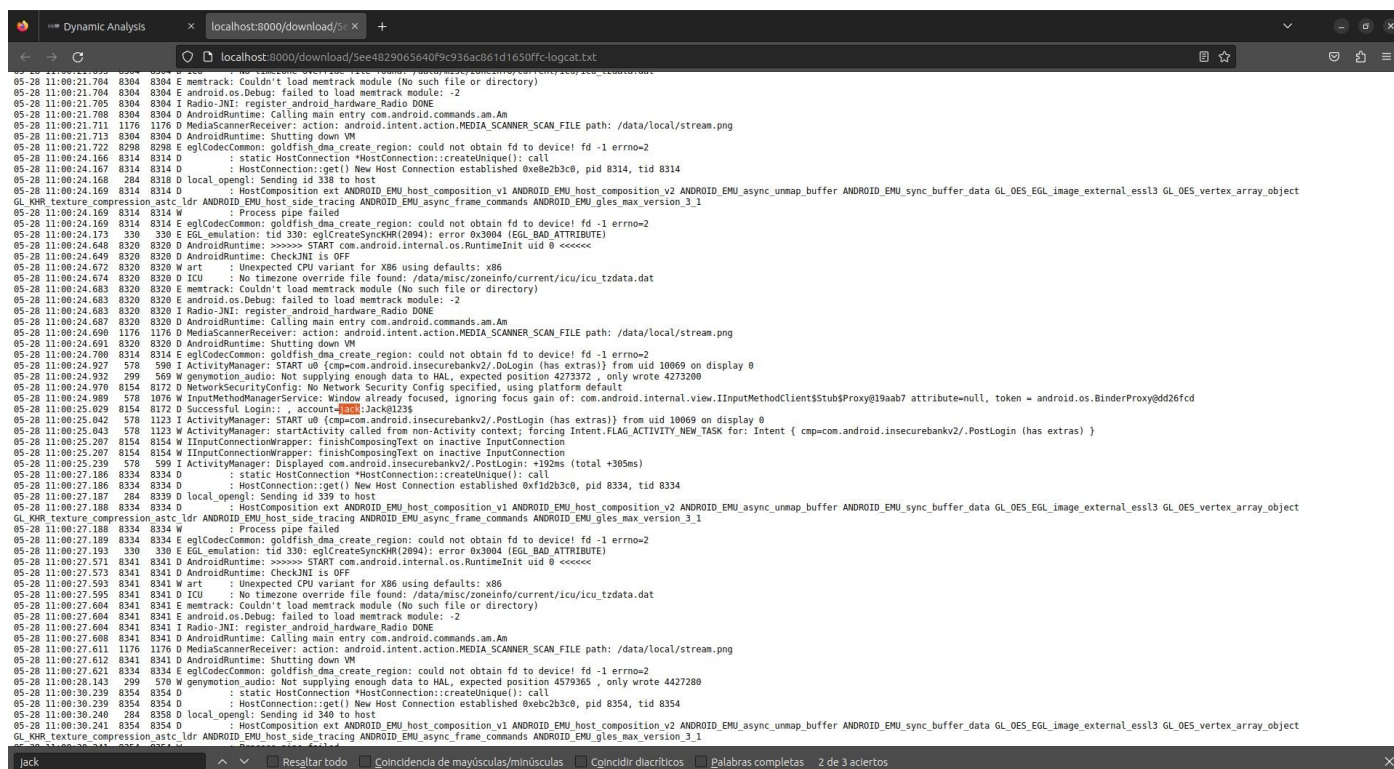
Si le damos a send y miramos en la aplicación de mensajería veremos que nos ha llegado un mensaje de conforme se ha realizado el cambio de contraseña.



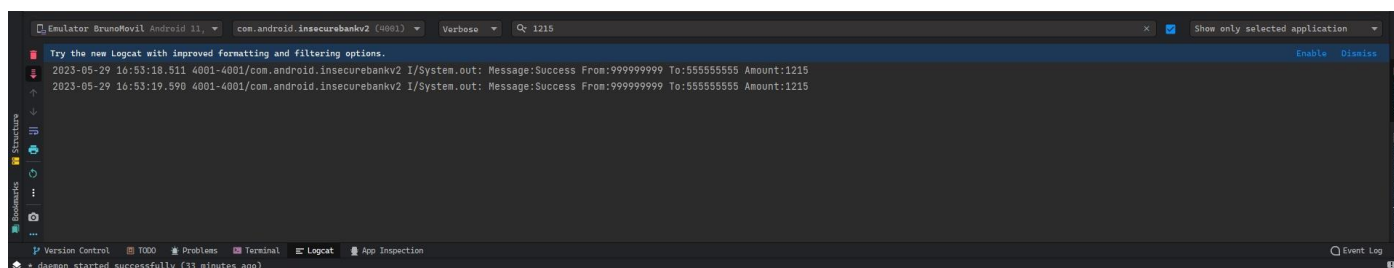
Como podemos ver aparece que se ha cambiado la contraseña. Todo esto es posible gracias a que el Broadcast esta en exported a true en el AndroidManifest.

LOGCAT DE LA APLICACION

En la herramienta de registro donde se muestran los mensajes del sistema, es posible que aparezca alguna credencial de algún usuario; si buscamos por el usuario “Jack” veremos que aparece tanto el nombre de usuario como la credencial en texto plano. A este log podemos acceder debido a que como vimos anteriormente el debug está activo en la aplicación.

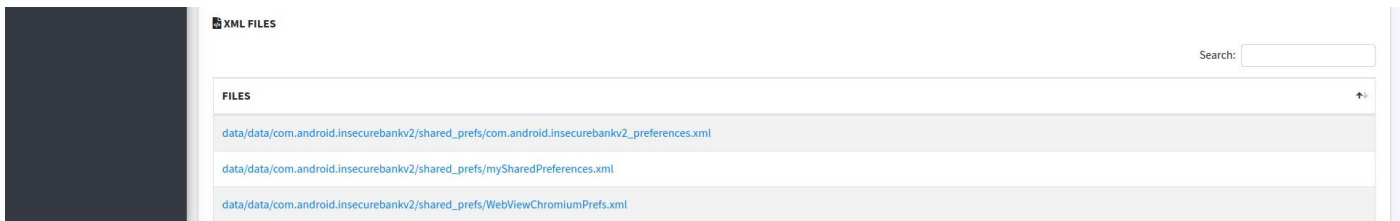


Aparte de que muestre el usuario y contraseña de una persona que ha hecho login; también muestra todas las transacciones realizadas.

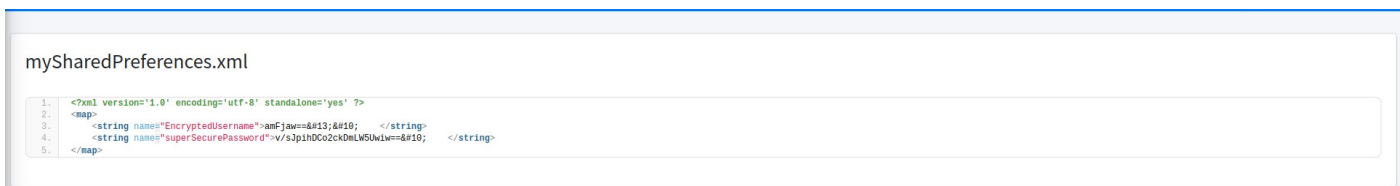


UNSECURE PASSWORD STORAGE

Como vimos anteriormente en el análisis estático había un método el cual permitía guardar las credenciales. Estas credenciales las guarda en un archivo llamado mySharedPreferences.xml. Con el análisis dinámico hemos conseguido acceso al archivo y hemos podido ver su contenido.

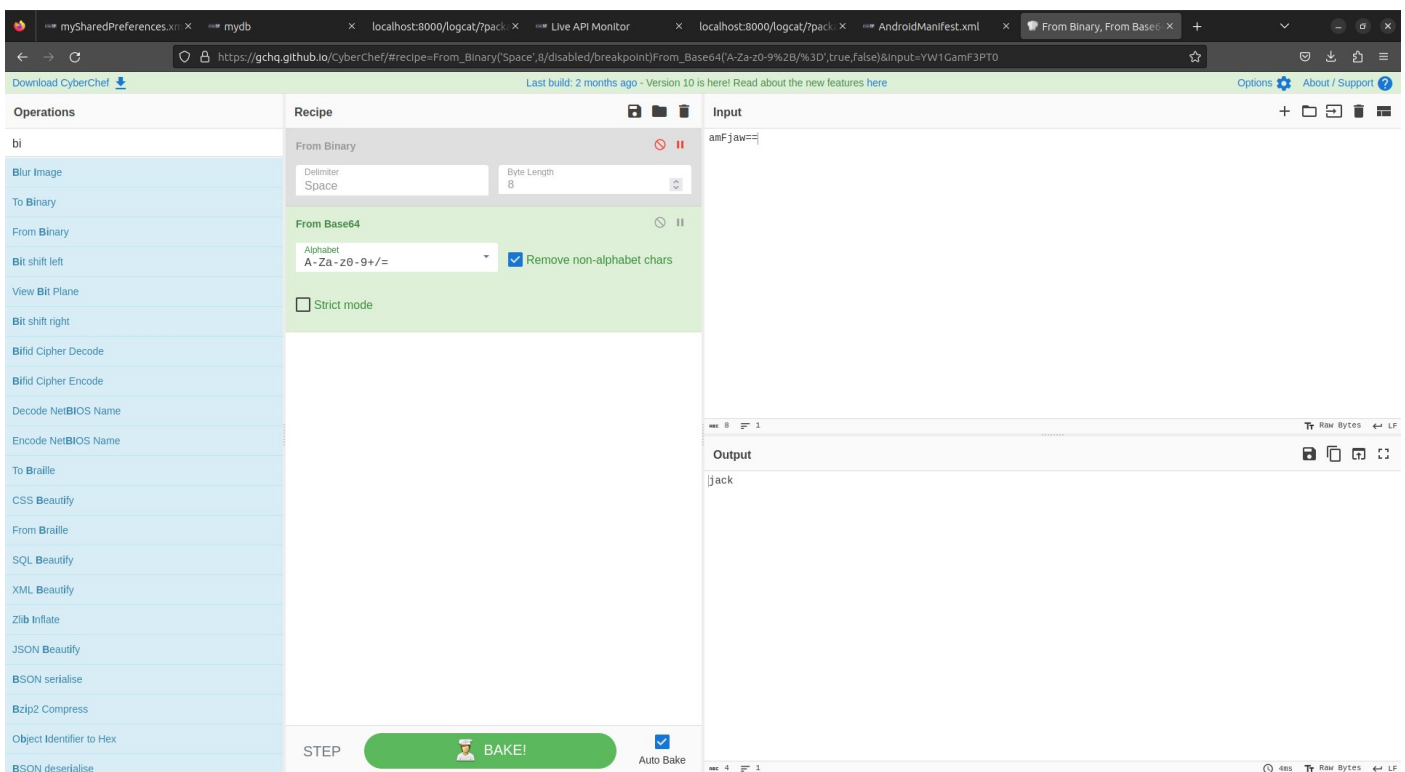


Si accedemos al archivo veremos lo siguiente.

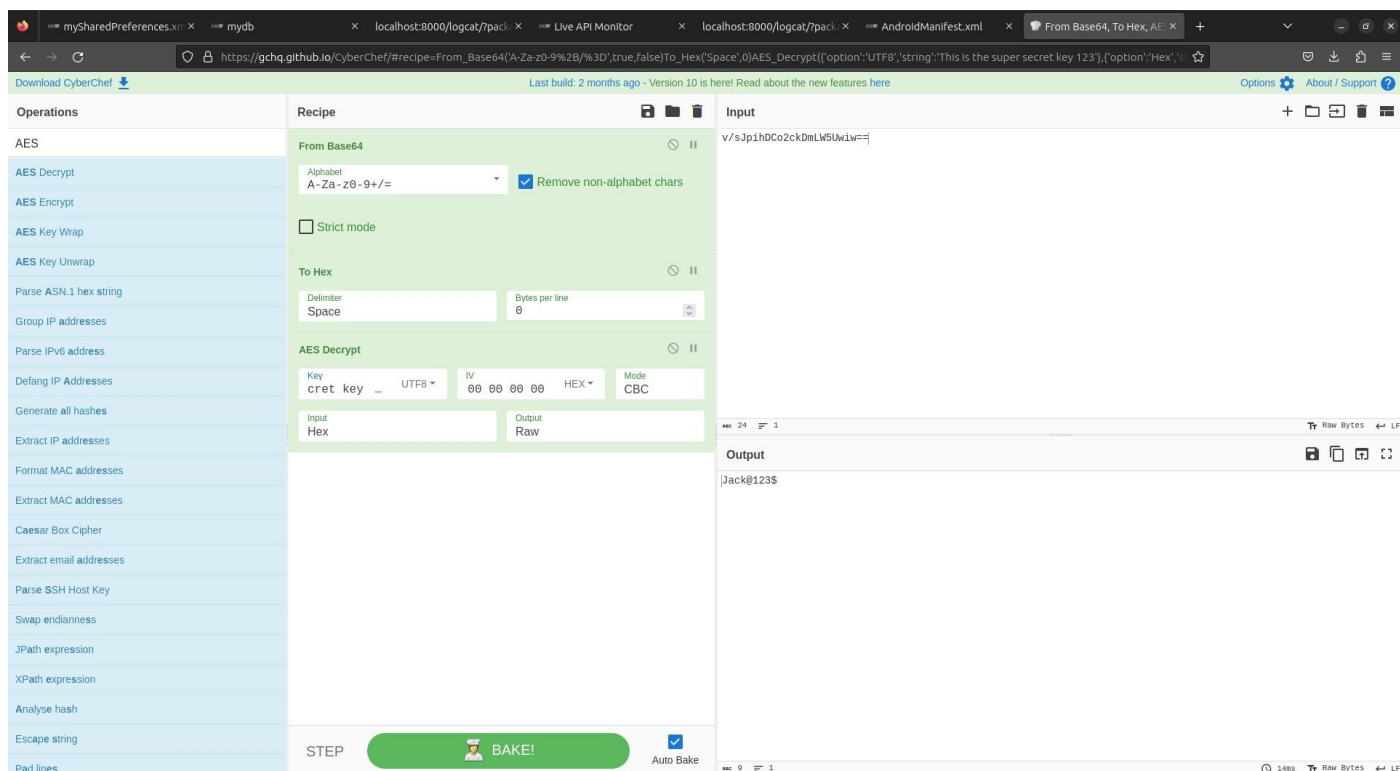


Como podemos ver aparece tanto el un usuario encriptado como una contraseña. Como vimos anteriormente el usuario estaba encriptado en base64 mientras que la contraseña estaba encriptada en AES. Si utilizamos Cyberchef para descriptarlos veremos tanto la contraseña como el usuario en texto plano.

Desencriptando usuario:



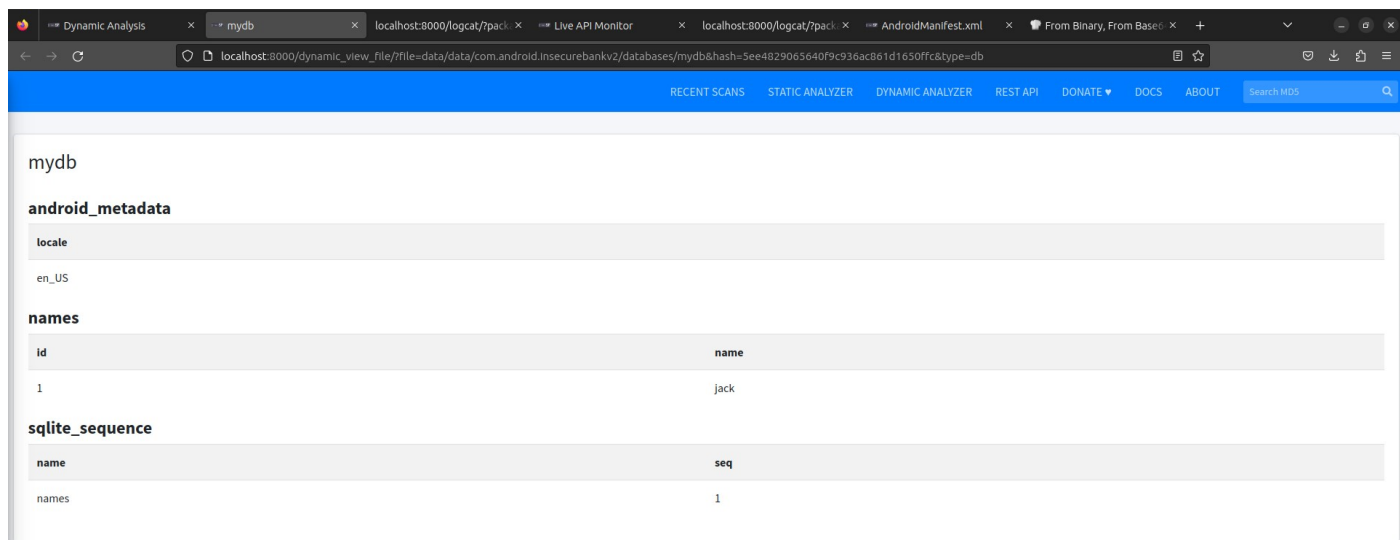
Desencriptando contraseña:



Como podemos ver en las capturas anteriores el usuario y la contraseña son jack y Jack@123\$ respectivamente.

ACCESS TO DB

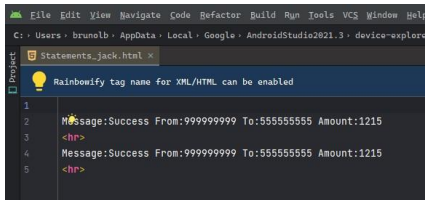
Gracias a que la aplicación tiene el backup activado podemos acceder a un archivo ,llamado mydb, durante el análisis dinámico. Si miramos el contenido del archivo veremos que contiene el usuario con el que se ha iniciado sesión en la aplicación.



UNSECURE DATA STORAGE

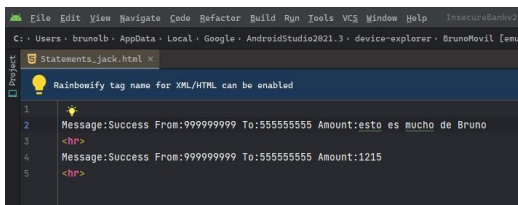
Como vimos antes se permite modificar la cantidad de las transacciones añadiendo el valor que nosotros queramos; por lo que podemos probar a modificarlo y ver lo que pasa.

Transferencia original:



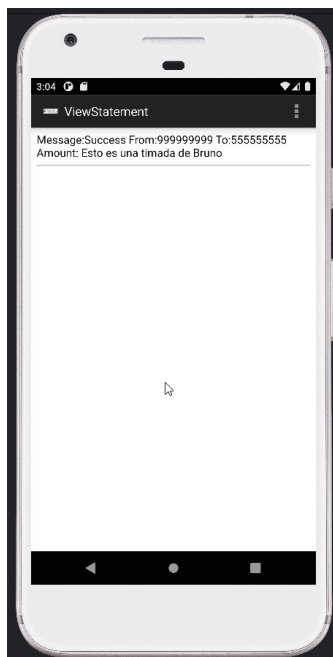
```
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
C:\Users\brunolb\AppData\Local\Google\AndroidStudio2021.3\device-explorer
Statements_jack.html x
Rainbowify tag name for XML/HTML can be enabled
1
2 Message:Success From:999999999 To:555555555 Amount:1215
3 <hr>
4 Message:Success From:999999999 To:555555555 Amount:1215
5 <hr>
```

Transferencia modificada:



```
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help InsureBankv2
C:\Users\brunolb\AppData\Local\Google\AndroidStudio2021.3\device-explorer\BrunoMovil [emu]
Statements_jack.html x
Rainbowify tag name for XML/HTML can be enabled
1
2 Message:Success From:999999999 To:555555555 Amount:esto es mucho de Bruno
3 <hr>
4 Message:Success From:999999999 To:555555555 Amount:1215
5 <hr>
```

Si ahora miramos en la aplicación en la opción de View Statement veremos como el valor ha cambiado.



INSECURE CONNECTIONS

Como hemos visto antes en el análisis estático todas las comunicaciones entre cliente y servidor se realizan a través de http, lo cual puede ser interceptado/modificado por un atacante. Que el atacante pueda modificarlo significa que podrá realizar peticiones automáticas para probar usuarios/contraseñas.

En nuestro caso usaremos wireshark para ver la comunicación.

15	13.262195296	192.168.1.57	192.168.1.89	HTTP	286 POST /login HTTP/1.1 (application/x-www-form-urlencoded)
16	13.262269165	192.168.1.89	192.168.1.57	TCP	54 8888 → 52769 [ACK] Seq=1 Ack=233 Win=64128 Len=0
17	13.268981360	192.168.1.89	192.168.1.57	TCP	189 8888 → 52769 [PSH, ACK] Seq=1 Ack=233 Win=64128 Len=135 [TCP ...
18	13.269425936	192.168.1.89	192.168.1.57	HTTP	104 HTTP/1.1 200 OK (text/html)
19	13.269667702	192.168.1.57	192.168.1.89	TCP	60 52769 → 8888 [ACK] Seq=233 Ack=186 Win=262400 Len=0
20	13.301880160	192.168.1.57	192.168.1.255	UDP	86 57621 → 57621 Len=44
21	16.007033844	MitraSta_e5:89:38	Broadcast	ARP	60 who has 192.168.1.40? Tell 192.168.1.1
22	16.417046338	192.168.1.1	224.0.0.1	IGMPv2	60 Membership Query, general

▶	Frame 15: 286 bytes on wire (2288 bits), 286 bytes captured (2288 bits) on interface enp0s3, id 0
▶	Ethernet II, Src: 5c:60:ba:33:43:18 (5c:60:ba:33:43:18), Dst: PcsCompu_32:40:d3 (08:00:27:32:40:d3)
▶	Internet Protocol Version 4, Src: 192.168.1.57, Dst: 192.168.1.89
▶	Transmission Control Protocol, Src Port: 52769, Dst Port: 8888, Seq: 1, Ack: 1, Len: 232
▶	Hypertext Transfer Protocol
▼	HTML Form URL Encoded: application/x-www-form-urlencoded
▶	Form item: "username" = "jack"
▶	Form item: "password" = "Jack@123\$"

Como podemos ver la contraseña y la password se transmiten a través de la red en texto plano. Esto se solucionaría poniendo el protocolo en https.

CORRECCION VULNERABILIDADES

BYPASSING LOGIN PAGE

Para corregir esta vulnerabilidad lo que debemos de hacer es poner el activity en exported=false en el AndroidManifest.xml, esto lo que hace es que no se pueda llamar a este activity desde otra aplicación. De paso que modificamos este activity modificaremos el resto de los activitys que tengan este error.

Código antes de corregir:

```
<activity
    android:name=".PostLogin"
    android:exported="true"
    android:label="PostLogin" >
</activity>
<activity
    android:name=".WrongLogin"
    android:label="WrongLogin" >
</activity>
<activity
    android:name=".DoTransfer"
    android:exported="true"
    android:label="DoTransfer" >
</activity>
<activity
    android:name=".ViewStatement"
    android:exported="true"
    android:label="ViewStatement" >
</activity>
```

```
<activity
    android:name=".ChangePassword"
    android:exported="true"
    android:label="ChangePassword" >
</activity>
```

Código corregido:

```
</activity>
<activity
    android:name=".PostLogin"
    android:exported="false"
    android:label="PostLogin" >
</activity>
<activity
    android:name=".WrongLogin"
    android:label="WrongLogin" >
</activity>
<activity
    android:name=".DoTransfer"
    android:exported="false"
    android:label="DoTransfer" >
</activity>
<activity
    android:name=".ViewStatement"
    android:exported="false"
    android:label="ViewStatement" >
</activity>
```

```
<activity
    android:name=".ChangePassword"
    android:exported="false"
    android:label="ChangePassword" >
</activity>
```

Como se puede ver en las capturas anteriores lo que hemos hecho es cambiar el `exported=true` a `exported=false` en todas las activities, esto impedirá que se llamen desde otros métodos.

LOGIN ADMIN

Esto permite iniciar sesión con un usuario llamado “devadmin”, esto es posible ya que en el código hay una parte que comprueba si se llama devadmin y en caso positivo inicia sesión, esto está en el archivo `DoLogin`.

Código antes de corregir:

```
if (username.equals("devadmin")) {
    httpPost2.setEntity(new UrlEncodedFormEntity(nameValuePair));
    // Execute HTTP Post Request
    responseBody = httpClient.execute(httpPost2);
} else {
    httpPost.setEntity(new UrlEncodedFormEntity(nameValuePair));
    // Execute HTTP Post Request
    responseBody = httpClient.execute(httpPost);
}
```

En este caso para que esto no ocurra lo que debemos de hacer es eliminar el `if` de la captura anterior y utilizar un sistema basado en roles de usuario.


```
nameValuePairs.add(new BasicNameValuePair("username", username));
nameValuePairs.add(new BasicNameValuePair("password", password));
HttpResponse responseBody;

InputStream in = responseBody.getEntity().getContent();
result = convertStreamToString( in );
result = result.replace( target: "\n", replacement: "");
if (result != null) {
    if (result.indexOf("Correct Credentials") != -1) {
        Log.d( tag: "Successful Login:", msg: " ", account=" + username + ":" + password);
        saveCreds(username, password);
        trackUserLogins();
        Intent pL = new Intent(getApplicationContext(), PostLogin.class);
```

El código una vez modificado quedaría de la siguiente manera; como podemos ver ya no se realiza la comprobación.

CREATE USER BUTTON

Esto lo que nos permite es mostrar un botón que permitiría la creación de un usuario cuando se implemente la funcionalidad. Esto lo realiza a través de un valor de un string, el cual se esta en el archivo strings.xml; en este caso no borraremos ninguna comprobación, solo comentaremos algunas líneas de código.

Código antes de corregir:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_log_main);
    String mess = "no";
    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);
    login_buttons.setOnClickListener((v) -> {
        // TODO Auto-generated method stub
        performLogin();
    });
    createuser_buttons = (Button) findViewById(R.id.button_CreateUser);
    createuser_buttons.setOnClickListener((v) -> {
        // TODO Auto-generated method stub
        createUser();
    });
}
```

```
1 usage  Dinesh Shetty +1
protected void createUser() {
    Toasteroid.show( activity: this, message: "Create User functionality is still Work-In-Progress!!", Toasteroid.STYLES.WARNING, Toasteroid.LENGTH_LONG);
}
}
```

Código corregido:

Lo que haremos en este caso es comentar todas las referencias al botón; también comentaremos el método que crea el usuario.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_log_main);
    String mess = "no";
    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);
    login_buttons.setOnClickListener((v) -> {
        // TODO Auto-generated method stub
        performlogin();
    });
    createuser_buttons = (Button) findViewById(R.id.button_CreateUser);
    // Dinesh Shetty +1 *
    createuser_buttons.setOnClickListener(new View.OnClickListener() {
        // Dinesh Shetty *
        @Override
        public void onClick(View v) {
            // TODO Auto-generated method stub
            createUser();
        }
    });
}
```

```
// protected void createUser() {
//     Toasteroid.show(this, "Create User functionality is still Work-In-Progress!!", Toasteroid.STYLES.WARNING, Toasteroid.LENGTH_LONG);
// }
// }
```

BROADCAST RECEIVER

Para corregir esta vulnerabilidad lo que debemos de hacer es poner el broadcast en exported=false en el AndroidManifest.xml, esto lo que hace es que no se pueda llamar a este broadcast desde otra aplicación.

Código antes de corregir:

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

Código corregido:

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

Como se puede ver en las capturas anteriores lo que hemos hecho es cambiar el exported=true a exported=false en el broadcast receiver, esto impedirá que se llamen desde otros métodos.

LOGCAT DE LA APLICACIÓN

Para que no se pueda acceder al logcat de una aplicación es debido a que la aplicación esta empaquetada y firmada con la firma de debug en lugar de estar empaquetada y firmada en modo release.

Para corregir esta vulnerabilidad lo que habría que hacer es re-empaquetar la aplicación en modo release.

UNSECURE PASSWORD STORAGE

Esta vulnerabilidad es que almacena el usuario y la contraseña en cifrados de poca seguridad. Para corregir esto debemos de utilizar funciones hash. Esto debemos de modificarlo en el archivo DoLogin.

```
private void saveCreds(String username, String password) throws UnsupportedOperationException, InvalidKeyException,
    // 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(), flags: 4));
    CryptoClass crypt = new CryptoClass();
    superSecurePassword = crypt.aesEncryptedString(rememberme_password);
    editor.putString(s: "EncryptedUsername", base64Username);
    editor.putString(s: "superSecurePassword", superSecurePassword);
    editor.commit();
}
```

Debemos de cambiar donde el cifrado por base64 y aes para el usuario y la contraseña respectivamente por una función hash del tipo sha256 o sha512.

ACCESS TO DB

Para resolver esta vulnerabilidad, lo que debemos de hacer es poner en AndroidManifest el valor de allowBackup de true a false. Aparte como en un caso anterior también debemos de empaquetar la aplicación en modo release.

Código sin corregir:

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="InsecureBankv2"
    android:theme="@android:style/Theme.Holo.Light.DarkActionBar">
    <!--
    android:theme="@style/AppTheme"-->
```

Código corregido:

```
<application
    android:allowBackup="false"
    android:icon="@mipmap/ic_launcher"
    android:label="InsecureBankv2"
    android:theme="@android:style/Theme.Holo.Light.DarkActionBar">
    <!--
    android:theme="@style/AppTheme"-->
```

Como podemos hemos desactivado el allowBackup en la aplicación. De esta forma ya no se podría acceder a la base de datos

CONTENT PROVIDER

Para corregir esta vulnerabilidad lo que debemos de hacer es poner el content provider en exported=false en el AndroidManifest.xml, esto lo que hace es que no se pueda llamar a este content provider desde otra aplicación.

Código antes de corregir:

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

Código corregido:

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

Como se puede ver en las capturas anteriores lo que hemos hecho es cambiar el exported=true a exported=false en el content provider, esto impedirá que se llamen desde otros métodos.

UNSECURE DATA STORAGE

El problema de esta vulnerabilidad es que almacena demasiada información en el log; para solucionarla debemos de comentar todas las líneas donde ponga log.d. Estas líneas se encuentran en los archivos: DoLogin,

Código sin corregir:

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

Código corregido:

```
InputStream in = responseBody.getEntity().getContent();
result = convertStreamToString( in );
result = result.replace( target: "\n", replacement: "" );
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( name: "uname", username );
        startActivity(pL);
    } else {
        Intent xi = new Intent(getApplicationContext(), WrongLogin.class);
        startActivity(xi);
    }
}
```

Esto desactivaría el almacenamiento inseguro de datos de la aplicación en el log.

INSECURE CONNECTIONS

Esta vulnerabilidad lo que permite es la captura del tráfico con las credenciales/información confidencial que se transmite entre servidor y cliente en texto plano. Para corregirla lo que debemos de hacer es poner la variable protocol con el string https://. Esto debemos de hacerlo en los archivos:

Código sin corregir:

```
2 usages
String protocol = "http://";
2 usages
```

Código corregido:

```
String serverPort = "8080";
2 usages
String protocol = "https://";
2 usages
```

Esto debemos de hacerlo para todas las coincidencias que encontremos en los archivos mencionados anteriormente.

EXECUTION JS ON VIEW STATEMENTS

Esto permite la ejecución de JS en la ventana donde se ven las transferencias; como vimos en el análisis dinámico es posible la modificación de ese fichero debido a que tiene permisos de escritura en la sdCard y este fichero se guarda en ella, en caso de que en el fichero mencionado se pusiese un código JS se ejecutaría al ver el statement.

Para solucionar esto debemos de prohibir la ejecución de JS en esta pantalla, esto debemos de hacerlo poniendo en false la opción.

Código sin corregir:

```
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);
```

Código corregido:

```
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);
```

Con esto ya no se podría ejecutar código JS en la pantalla de ViewStatements.

Bruno López Barcia

REPOSITORIO GITHUB

[GitPPSBruno/InsecureBankv2Corregido \(github.com\)](https://github.com/GitPPSBruno/InsecureBankv2Corregido)