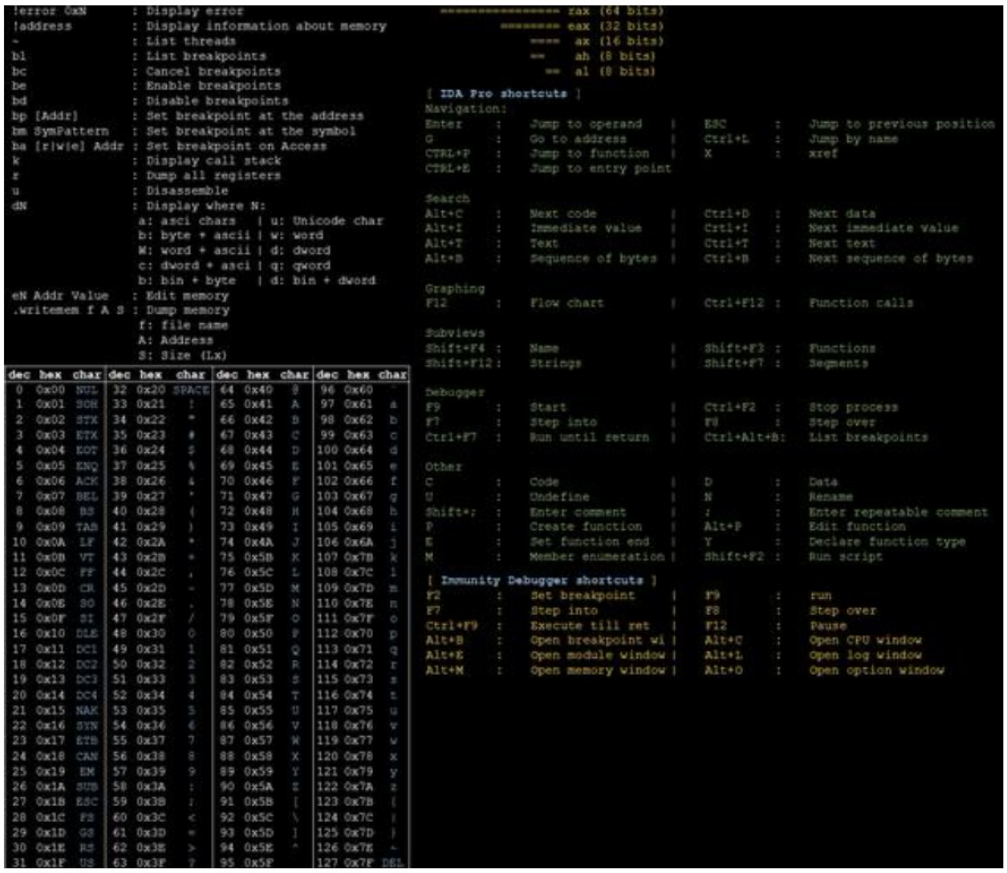
REVERSING DE SISTEMAS OPERATIVOS MOVILES



Máster en Análisis de Malware, Reversing y Bug Hunting

Ramon González Gaztelupe

1.- Análisis Estático "InsecureBank v2"

Analizando el *.xml AndroidManifest* podemos observar como es posible que se pueda realizar una llamada directa a la actividad PostLogin y saltarnos la actividad que nos pide las credenciales en la aplicación:

Tenemos la propiedad ***android:exported="true"*** lo que es una vulnerabilidad en este caso, ya que no deberíamos de poder iniciarla desde cualquier lugar:

- Si es **true**, cualquier app puede acceder a la actividad e iniciarla por su nombre exacto de clase.
- Si es **false**, solo los componentes de la misma aplicación, las aplicaciones con el mismo ID de usuario o los componentes del sistema con privilegios pueden iniciar la actividad.

```
<activity android:label="@string/title_activity_file_pref" android:name="com.android.insecurebankv2.FilePrefActivity" android:windowSoftInputMode="adjustNothing|stateV
<activity android:label="@string/title_activity_do_login" android:name="com.android.insecurebankv2.DoLogin"/>
<activity android:label="@string/title_activity_post_login" android:name="com.android.insecurebankv2.PostLogin" android:exported="true"/>
<activity android:label="@string/title_activity_wrong_login" android:name="com.android.insecurebankv2.WrongLogin" />
<activity android:label="@string/title_activity_do_transfer" android:name="com.android.insecurebankv2.DoTransfer" android:exported="true"/>
<activity android:label="@string/title_activity_view_statement" android:name="com.android.insecurebankv2.ViewStatement" android:exported="true"/>
<provider android:name="com.android.insecurebankv2.TrackUserContentProvider" android:exported="true" android:authorities="com.android.insecurebankv2.TrackUserContentPr
<receiver android:name="com.android.insecurebankv2.MyBroadCastReceiver" android:exported="true">
```

Si visualizamos el código en DoLogin que seria la actividad que nos debería transportar a PostLogin después de verificar nuestras credenciales, comprobamos que se realiza:

- **startActivity(pL)**
- **pL = new Intent() → PostLogin.class**

Por lo tanto una vez verificadas las credenciales mediante HTTP/JSON (*que por cierto no se usa TLS para el envío de datos*):

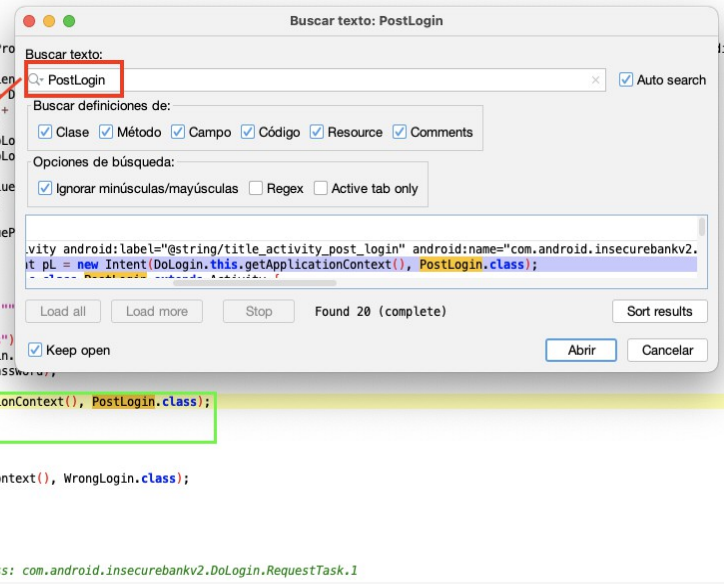
```
if (DoLogin.this.result != null) {
    if (DoLogin.this.result.indexOf("Correct Credentials") != -1) {
```

Iniciará una nueva actividad transportándoos a nueva clase llamada **PostLogin.class**:

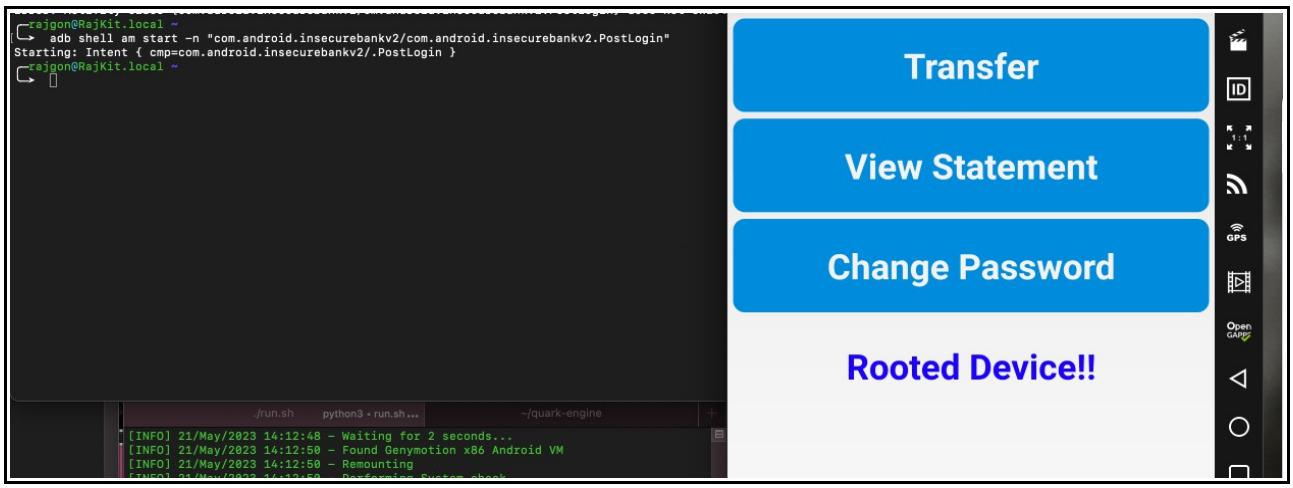
```
protected void onProgressUpdate(Integer... progress) {
}

public void postData(String valueIWantToSend) throws ClientPro
    HttpResponse responseBody;
    DefaultHttpClient defaultHttpClient = new DefaultHttpClien
    HttpPost httppost = new HttpPost(DoLogin.this.protocol +
    HttpPost httppost2 = new HttpPost(DoLogin.this.protocol +
    List<NameValuePair> nameValuePairs = new ArrayList<>(2);
    nameValuePairs.add(new BasicNameValuePair("username", DoLo
    nameValuePairs.add(new BasicNameValuePair("password", DoLo
    if (DoLogin.this.username.equals("devadmin")) {
        httppost2.setEntity(new UrlEncodedFormEntity(nameValue
        responseBody = defaultHttpClient.execute(httppost2);
    } else {
        httppost.setEntity(new UrlEncodedFormEntity(nameValueP
        responseBody = defaultHttpClient.execute(httppost);
    }
    InputStream in = responseBody.getEntity().getContent();
    DoLogin.this.result = convertStreamToString(in);
    DoLogin.this.result = DoLogin.this.result.replace("\n", ""
    if (DoLogin.this.result != null) {
        if (DoLogin.this.result.indexOf("Correct Credentials")
            Log.d("Successful Login:", ", account=" + DoLogin.
            saveCreds(DoLogin.this.username, DoLogin.this.passwo
            trackUserLogins();
            Intent pL = new Intent(DoLogin.this.getApplicationContext(), PostLogin.class);
            pL.putExtra("uname", DoLogin.this.username);
            DoLogin.this.startActivity(pL);
            return;
        }
        Intent xi = new Intent(DoLogin.this.getApplicationContext(), WrongLogin.class);
        DoLogin.this.startActivity(xi);
    }
}

private void trackUserLogins() {
    DoLogin.this.runOnUiThread(new Runnable() { // from class: com.android.insecurebankv2.DoLogin.RequestTask.1
```



Lo que significa que podemos acceder directamente llamando a la actividad mediante **“am start -n”**

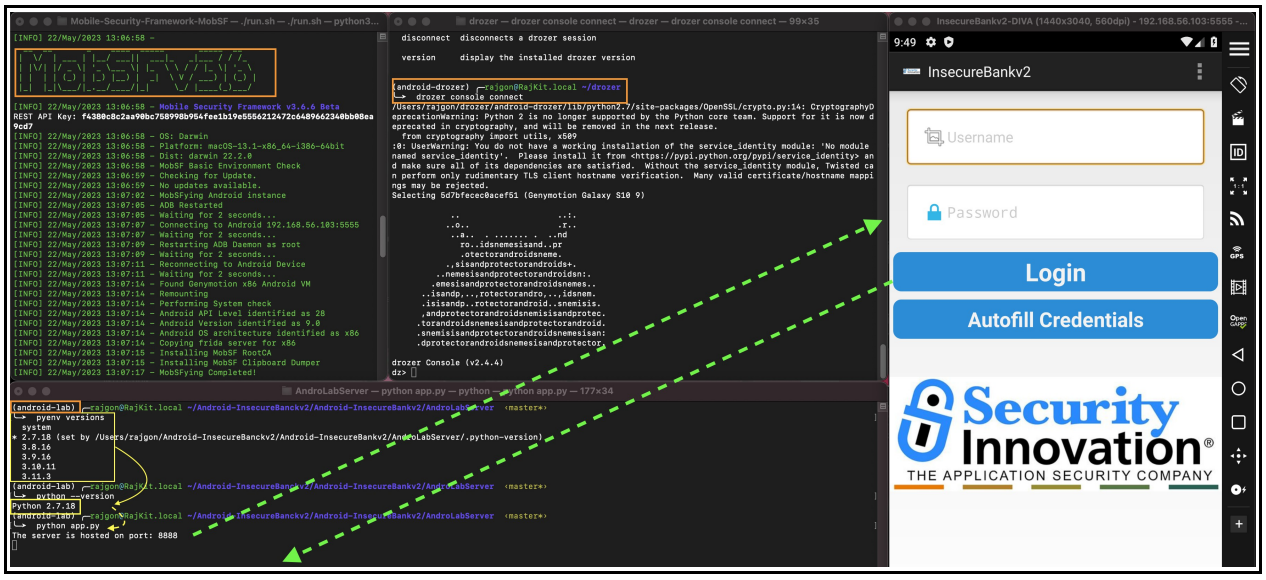


2.- Análisis Dinámico "InsecureBank v2"

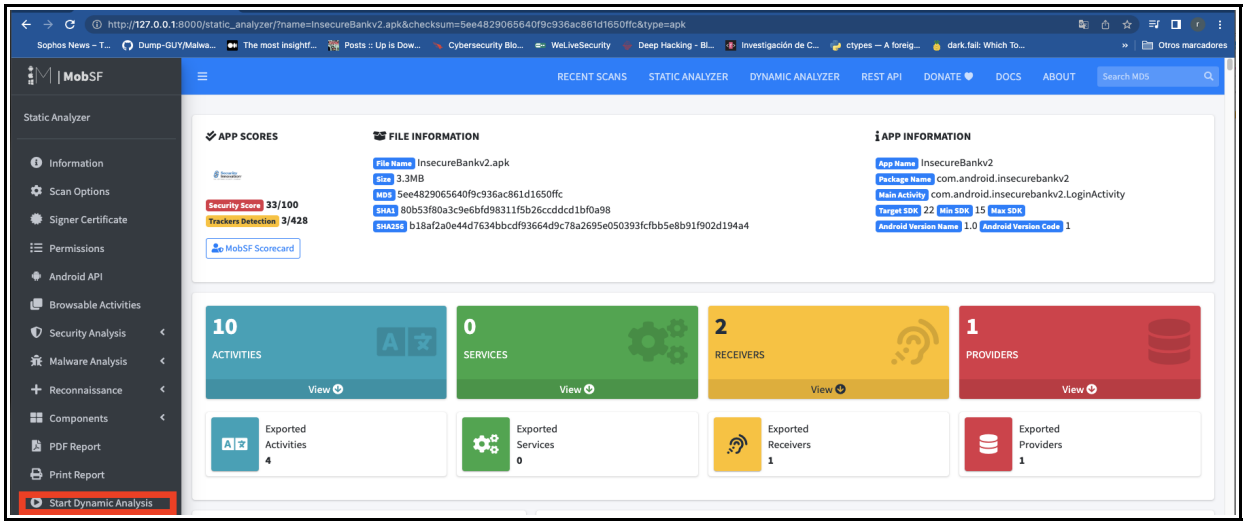
Lo primero que tendremos que hacer es configurar el entorno de análisis y arrancar el back-end de la aplicación **InsecureBankv2**, que será el que reciba procese las peticiones POST de la apk para devolvernos un JSON con la respuesta del servidor al **request**:

Para arrancar el back-end hemos creado un entorno virtual para cuidar las dependencias y hemos elegido la versión de *python 2.7.18*, ya que el script corre con *python 2*:

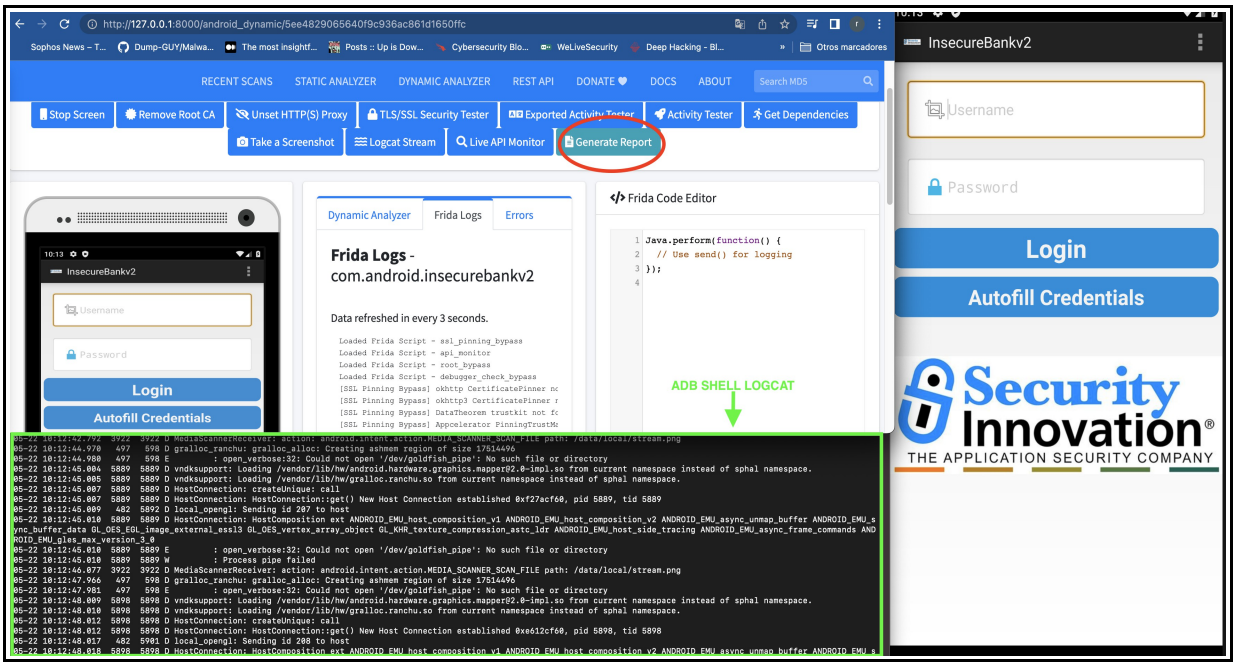
Aunque tengamos *Frida*, también dejare una conexión con *Drozer* por si nos viniese útil durante el dinámico:



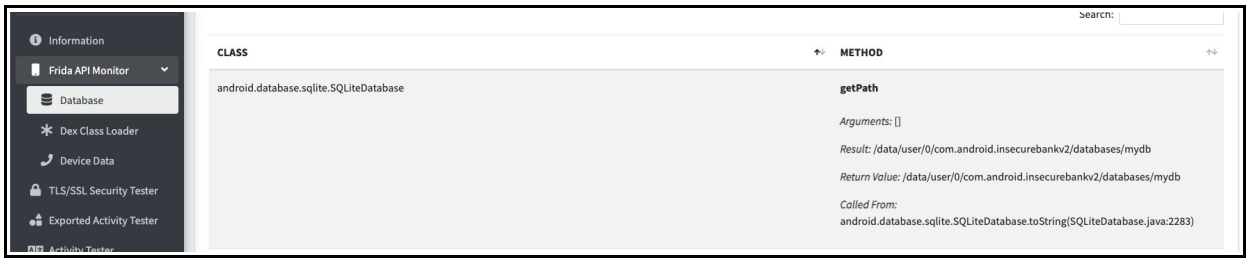
Hacemos un análisis estático con *MobSF*, drag & drop y arrancamos el dinámico desde el resultado:



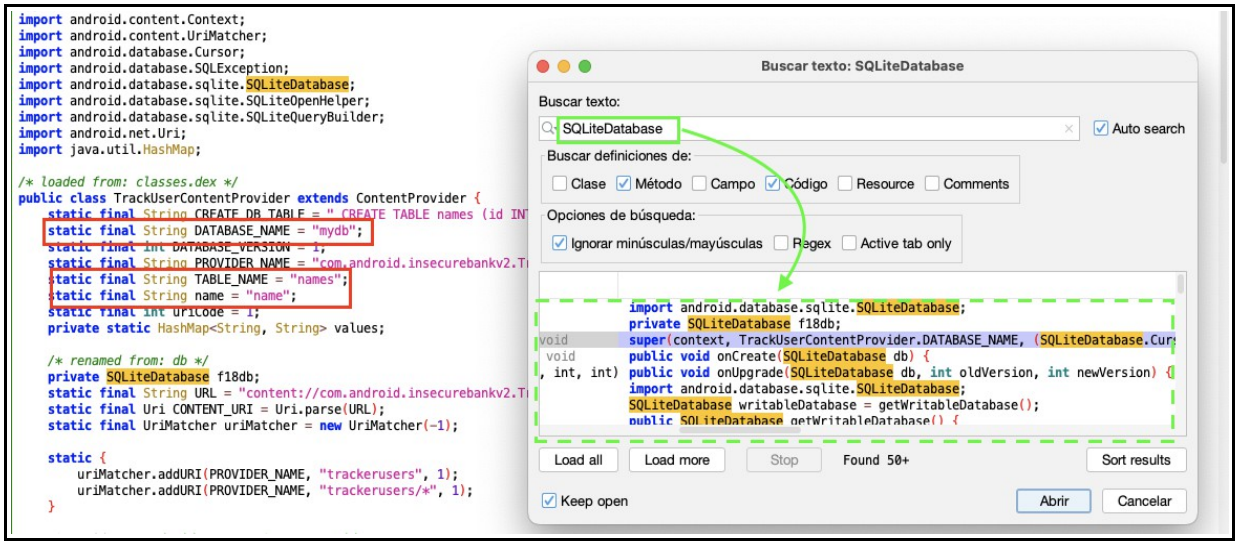
Generamos el reporte a partir del análisis dinámico que a realizado *MobSF*:



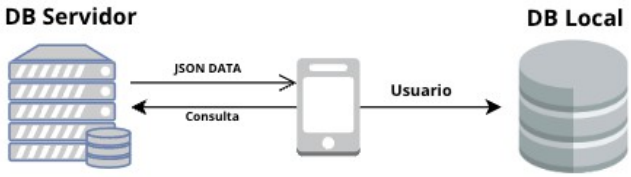
Vemos que se utiliza la clase ***SQLiteDatabase*** para realizar consultas SQL:



Realizamos una búsqueda a través *jadx-gui* y nos lleva a la clase **TrackUserContentProvider**, una clase propia de la aplicación la cual contiene variables estáticas con el nombre de la base de datos a la que se accede:



Por lo tanto la aplicación se comunica con el servidor, comprueba las credenciales y si son correctas almacena localmente el usuario:



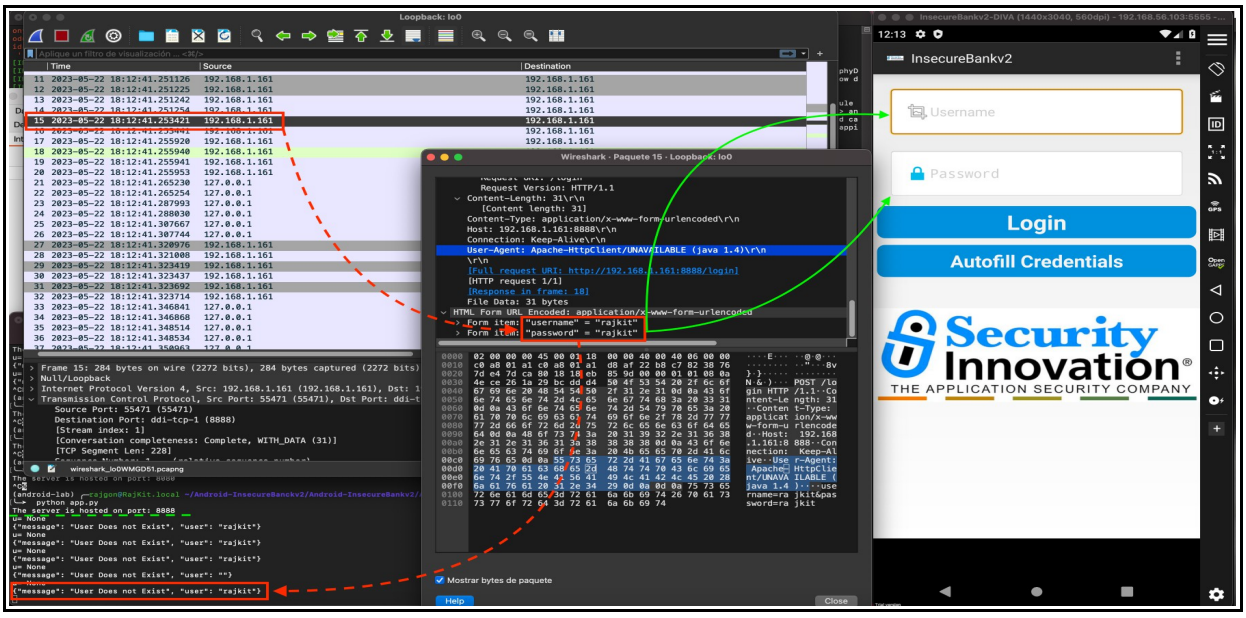
Realizamos una consulta en el servidor de forma local para ver que usuarios tenemos y probar la consulta desde la aplicación.

Utilizando las credenciales **USER:** jack **PASSWORD:** Jack@123$, obtenemos lo siguiente:

```
(android-lab) ┌─rajgon@RajKit.local ~/Android-InsecureBanckv2/Android-InsecureBankv2/AndroLabServer ‹master*›
└─➤  python app.py --port 8899
The server is hosted on port: 8899
u= None
{"message": "User Does not Exist", "user": ""}
u= <User u'jack'>
{"message": "Wrong Password", "user": "jack"}
u= <User u'jack'>
{"message": "Correct Credentials", "user": "jack"}
```

Servidor	Aplicación
`└─➤  sqlite3 mydb.db` `SQLite version 3.39.5 2022-10-14 20:58:05` `Enter ".help" for usage hints.` `sqlite> .tables` `accounts  users` `sqlite> select * from users;` `1|dinesh|Dinesh@123$\|dinesh\|shetty` `2\|jack\|Jack@123$|jack|apples`	`sqlite> select * from names;` `id  name` `----------  ----------` `1  jack` `sqlite>`

Utilizaremos *Wireshark* durante la consulta de credenciales entre la aplicación y el servidor, configuraremos un proxy como pantalla entre el dispositivo móvil y el server:



Si la aplicación hubiese usado ***TLSoverHTTP*** podríamos instalar certificados propios en el dispositivo y con un *BurpSuite* configurar un proxy local para hacer *MitM*, tendríamos que *bypassear* el ***SSL Pinning*** , que se le denomina al proceso de verificar que el certificado sea el propio del servidor de la aplicación y no de otro cualquiera.

Para realizar eso tendríamos que hacer un *hook* con *Frida* inyectando este script:

```
┌─rajgon@RajKit.local ~/Downloads
└─➤  cat frida-ssl-bypass.js
/*
   Universal Android SSL Pinning Bypass
   by Mattia Vinci and Maurizio Agazzini

   $ frida -U -f org.package.name -l universal-ssl-check-bypass.js --no-pause

    https://techblog.mediaservice.net/2018/11/universal-android-ssl-check-bypass-2/
*/

Java.perform(function() {

    var array_list = Java.use("java.util.ArrayList");
    var ApiClient = Java.use('com.android.org.conscrypt.TrustManagerImpl');

    ApiClient.checkTrustedRecursive.implementation = function(a1, a2, a3, a4, a5, a6) {
        // console.log('Bypassing SSL Pinning');
        var k = array_list.$new();
        return k;
    }

}, 0);
```

Lo inyectaríamos con este comando a través del *./**frida-server*** previamente instalado en el dispositivo Android:

- ***frida -U -f com.android.insecurebankv2 -l frida-ssl-bypass.js -o-pause***