Ciberseguridad en dispositivos móviles

Práctica 6

Luis López Cuerva
Pablo Alcarria Lozano

Luis López Cuerva
Pablo Alcarria Lozano

Introducción

Durante esta práctica se van a revisar las anteriores fallas de seguridad encontradas en las prácticas, repasando el problema causado y una posible corrección del problema o la vulnerabilidad. En general, muchos de los problemas se pueden solucionar eliminado código que hacía más fácil de romper la aplicación y recortando los permisos necesarios a los mínimos que requiere la aplicación.

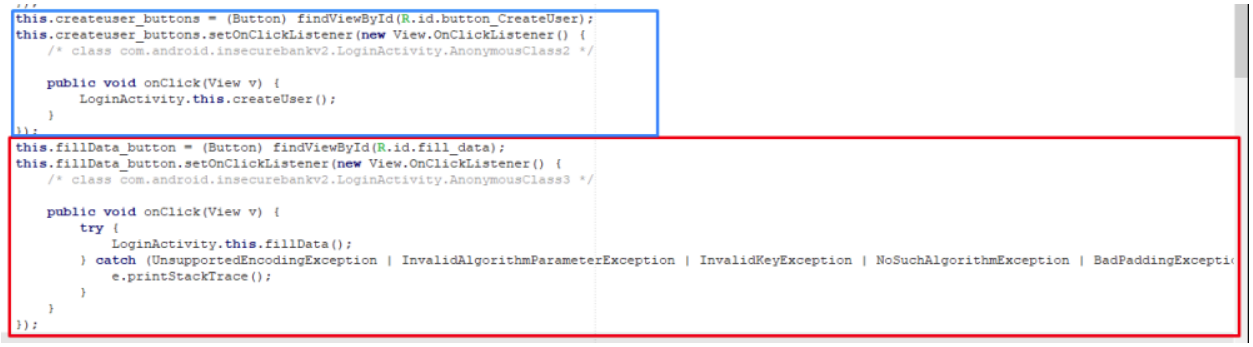
Configuración de componentes de la aplicación

Durante el análisis de la aplicación, podemos encontrar que las aplicaciones puedan ser iniciadas a través de componentes de otras aplicaciones ya que en las etiquetas de las actividades están calificadas como “exported=true” en el fichero AndroidManifest.xml. Teniendo en cuenta la funcionalidad de las actividades, no es necesario que tengan el atributo “exported” en ninguna. También podemos ver que está el atributo “allowBackup=true”, por lo que se puede hacer una copia de seguridad con los datos de la aplicación, por lo que si se consigue hacer un backup incluyendo la parte de shared prefs, tendría acceso a las credenciales del usuario del dispositivo, y aunque estén cifradas ya hemos demostrado en anteriores prácticas cómo desencriptarlas. Por último, el “modo debugging” le hace la vida más fácil a un posible atacante, de forma que puede conocer con más precisión el funcionamiento del programa y el flujo de ejecución, así que desactivarlo dentro del fichero AndroidManifest.xml que incluya la aplicación final que se vaya a lanzar, también sería recomendable.

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-feature android:glEsVersion="0x00020000" android:required="true"/>
<application android:allowBackup="true" android:debuggable="true" android:exported="true" android:icon="@mipmap/ic_launc
    <activity android:label="@string/app_name" android:name="com.android.insecurebankv2.LoginActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN"/>
            <category android:name="android.intent.category.LAUNCHER"/>
        </intent-filter>
    </activity>
    <activity android:label="@string/title_activity_file_pref" android:name="com.android.insecurebankv2.FilePrefActivity
    <activity android:label="@string/title_activity_do_login" android:name="com.android.insecurebankv2.DoLogin"/>
    <activity android:exported="true" android:label="@string/title_activity_post_login" android:name="com.android.insecu
    <activity android:label="@string/title_activity_wrong_login" android:name="com.android.insecurebankv2.WrongLogin"/>
    <activity android:exported="true" android:label="@string/title_activity_do_transfer" android:name="com.android.insec
    <activity android:exported="true" android:label="@string/title_activity_view_statement" android:name="com.android.in
    <provider android:authorities="com.android.insecurebankv2.TrackUserContentProvider" android:exported="true" android:
    <receiver android:exported="true" android:name="com.android.insecurebankv2.MyBroadCastReceiver">
        <intent-filter>
            <action android:name="theBroadcast"/>
        </intent-filter>
    </receiver>
    <activity android:exported="true" android:label="@string/title_activity_change_password" android:name="com.android.i
    <activity android:configChanges="keyboard|keyboardHidden|orientation|screenLayout|screenSize|smallestScreenSize|uiMo
    <activity android:name="com.google.android.gms.ads.purchase.InAppPurchaseActivity" android:theme="@style/Theme.IAPTh
    <meta-data android:name="com.google.android.gms.version" android:value="@integer/google_play_services_version"/>
    <meta-data android:name="com.google.android.gms.wallet.api.enabled" android:value="true"/>
    <receiver android:exported="false" android:name="com.google.android.gms.wallet.EnableWalletOptimizationReceiver">
```

Botón oculto

Dentro de la aplicación, en la actividad principal LoginActivity.java nos encontramos un botón oculto, ya comentado en anteriores prácticas. Este botón, aunque no incluye ninguna funcionalidad y sólo muestra un toast en la pantalla cuando está activado, podemos aprovecharlo con Frida para bypassear la pantalla de login mediante un intent nuevo, o cualquier cosa que queramos inyectar en ella.



```
this.createuser_buttons = (Button) findViewById(R.id.button_CreateUser);
this.createuser_buttons.setOnClickListener(new View.OnClickListener() {
    /* class com.android.insecurebankv2.LoginActivity.AnonymousClass2 */

    public void onClick(View v) {
        LoginActivity.this.createUser();
    }
});
this.fillData_button = (Button) findViewById(R.id.fill_data);
this.fillData_button.setOnClickListener(new View.OnClickListener() {
    /* class com.android.insecurebankv2.LoginActivity.AnonymousClass3 */

    public void onClick(View v) {
        try {
            LoginActivity.this.fillData();
        } catch (UnsupportedEncodingException | InvalidAlgorithmParameterException | InvalidKeyException | NoSuchAlgorithmException | BadPaddingExceptio
            e.printStackTrace();
        }
    }
});
```

Las buenas prácticas marcan que este botón (rodeado en azul en la imagen) no debería existir, ya que es una potencial vulnerabilidad. Hay que intentar que un posible atacante lo tenga lo más difícil posible, y eliminando el botón eliminamos un vector de ataque.

Usuario devadmin

Dentro del propio código de la aplicación, en el método doLogin.java nos encontramos que existe un usuario “secreto” en la aplicación en la que no se solicita una contraseña, creando una petición HTTP distinta a la creada por usuarios normales. Este código es visible para cualquier usuario que decompile la aplicación y lea el código, así que debe ser inmediatamente eliminado también para evitar accesos no deseados a la aplicación.

```
public void postData(String valueIWantToSend) throws ClientProtocolException, IOException, JSONException, InvalidKeyException, NoSuchAl
    HttpResponse responseBody;
    HttpClient httpclient = new DefaultHttpClient();
    HttpPost httppost = new HttpPost(DoLogin.this.protocol + DoLogin.this.serverip + ":" + DoLogin.this.serverport + "/login");
    HttpPost httppost2 = new HttpPost(DoLogin.this.protocol + DoLogin.this.serverip + ":" + DoLogin.this.serverport + "/devlogin");
    List<NameValuePair> nameValuePairs = new ArrayList<>(2);
    nameValuePairs.add(new BasicNameValuePair("username", DoLogin.this.username));
    nameValuePairs.add(new BasicNameValuePair("password", DoLogin.this.password));
    if (DoLogin.this.username.equals("devadmin")) {
        httppost2.setEntity(new UrlEncodedFormEntity(nameValuePairs));
        responseBody = httpclient.execute(httppost2);
    } else {
        httppost.setEntity(new UrlEncodedFormEntity(nameValuePairs));
        responseBody = httpclient.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) {
        return;
```

Shared preferences

La aplicación permite almacenar de forma local las credenciales de inicio de sesión, siendo una aplicación bancaria dicha opción no es segura. Para solucionar este problema se han seguido los pasos indicados en la práctica. Entre estos pasos nos gustaría destacar el cambio de ubicación de almacenamiento. Originalmente este archivo guardaba la información en la SD, donde cualquier aplicación podía acceder, posteriormente se guarda en el espacio de almacenamiento interno de la aplicación.

Caché del teclado

La aplicación hace un uso incorrecto de los campos de texto de la interfaz de usuario y por lo tanto el dispositivo móvil realiza una copia caché para mejorar el autocorrector del dispositivo.

Para solucionar este problema se propone convertir todos los campos de texto en los que se introduzca información de inicio de sesión en campos tipo *password*, cuyo contenido no se guarda para mejorar el autocorrector. Para los campos que deban ser visibles se sugiere simplemente activar la opción que permite leer el contenido del campo.

Broadcast receiver

La aplicación permite el cambio de credenciales de una forma insegura, ya que envía por un sms la contraseña antigua y la nueva, además, para realizar esta modificación utiliza un broadcast receiver que se puede activar por parte de otras aplicaciones. Para solucionar este hecho se propone cambiar la configuración de exported, cosa que ya se ha hecho en el primer apartado y modificar esta funcionalidad, de manera que informe que ha habido un cambio de credenciales pero sin informar de las credenciales anteriores o las nuevas.

Concretamente se propone eliminar las líneas enmarcadas en rojo y modificar el código existente entre las líneas destacadas en azul, ambas incluidas, de manera que el sms que envíe la aplicación resulte en el siguiente: “Se ha producido un cambio de credenciales.”.

Encriptación insuficiente

Si bien la aplicación realiza una encriptación de los datos esta encriptación es completamente inútil ya que el vector de inicialización y el vector de cifrado están hardcodeados en el código, visibles en la clase java **CryptoClass**. Para que esta encriptación aumentase la dificultad de robo de credenciales estos dos valores deberían ser secretos.

Conclusiones

Una vez vistas las abundantes deficiencias en materia de seguridad que presenta la aplicación se pone de manifiesto la importancia que tiene seguir un ciclo de desarrollo que tenga en cuenta la seguridad. En el estado actual la aplicación se puede parchear para hacerla más segura, sin embargo nunca llegará a ser tan segura como si desde la fase de obtención de requisitos se hubiera tenido en cuenta la seguridad.