uc3m | Universidad Carlos III de Madrid
Grupo de investigación:
Computer Security Lab

Miguel Ángel Díaz Bautista
Roberto Blanco Ancos

Module III Assignment: Secure storage of persistent data

Mobile Devices Security

Table of Contents

1 Introduction

Even though the majority of users are not aware of the necessity of encypting their devices, corporate users doubt and fear that their data might get compromised in case of the loss or robbery of the device.

In this 3rd module, we describe some of the techniques that are generally used to encrypt sensitive data in our mobile devices as well as to protect data in transit, avoiding that such data ends in unauthorized hands.

It will also help us analyze and use cryptographic libraries specific for the secure storage of persistent data from any Android application. We will also cover how to correctly use the encryption and avoid some of the most common antipatterns to keep the data secure inside our Android app. We will recommend some 3rd party libraries already implemented and tested to save time and to help us rapidly increase the security of the database of any application.

The objectives for the assignment are:

- Usage of the cryptographic libraries for Android
- Understand and generate symmetric encryption keys
- Understand and use the Android KeyStore provider
- Understand and use the encryption of a database using SQLCipher
- Enhance the level of trust on connections made to remote services, by means of HTTPS and *Certificate Pinning* (also known as *Public Key Pinning*, depending on how we use it)

In this module we will be using:

- SQLCipher: https://www.zetetic.net/sqlcipher/

2 Encryption of the *CredHub* app database

The *CredHub* app already implemented in previous assignments must be modified so the data stored in its database is secure.

The app requisites are :

- The app must retain the functionality described in the first assignment.
- It must retain the login screen and HTTP Basic authentication, both introduced in the second assignment.
- Must have the user's data database encrypted with SQLCipher, using a random symmetric key that is protected via the Android KeyStore.

When encrypting sensitive data using the KeyStore Provider we need:

- A PIN code that unlocks the mobile device once the user wants to use it.
- Create the Android KeyStore storage
- Create a pair of keys, private and public. When creating it, an alias is provided, which is needed to load the keys later on.
- When creating the database we create a password (random) only once, and this will be encrypted with the public key generated in previous steps. The key will be stored locally.
- Transparent encryption/decryption of the database<using SQLCipher, using the private key once the alias has been given.

When the modifed app with its ciphered database is complete, the corresponding .DB file must be extracted in order to verify that, now that it is ciphered, it cannot be analyzed from outside the Android device.

3 HTTPS and *Certificate Pinning*

Now we will add yet another modification to the *CredHub* app, this time to protect the data transmitted by the app over the network.

The tasks to be performed in this phase are the following:

- Use HTTPS instead of HTTP for all connections to the web server: HTTPS allows transparent negotiation of secure channels through TLS (*Transport Layer Security*) protocol.
- Analyze once again the communications between app and webservice with Wireshark, the same way we did during Module II assignment, and check whether or not data is indeed ciphered this time around. Take note of the certificate sent by the web server to the app.
- Incorporate *Certificate Pinning* functionalities in order to enhance the level of trust in the identity of the remote web server.

We must keep in mind that, for the web server to be able to manage HTTPS connections, the server´s executable file must be launched using the following input arguments:

```
"%JAVA_HOME%\bin\java" -jar SDM_WebRepo.jar https+auth
```

When implementing *certificate pinning* in the application, we will make use of the **`android:networkSecurityConfig`** attribute in the manifest file of the app. This makes the next two tasks much easier:

- Defining a `<pin-set>` which contains the SHA-256 hashes of the public keys in those certificates that the app will automatically accept. Only those connections with servers hosting certificates included in this pin-set will be established successfully.
- Including external root certificates, different from those defined in the system by default, through the `<trust-anchors>` property. A file *sdm_ca.cer* will be provided to the students, containing a self-signed CA certificate used by the web server in the remote repository, in order to be included in this section.

In order to verify the effectiveness of *certificate pinning* against a *'phishing'* or *'man-in-the-middle'* attack, the web server may be launched in a special operation mode that makes it publish a rogue certificate:

```
"%JAVA_HOME%\bin\java" -jar SDM_WebRepo.jar https+auth+rogue
```

4 Assignment delivery

This assignment will be carried out in groups of two students, the same groups assigned in previous assignments.

The assignment will be delivered through Aula Global :

- Report in pdf format that describes the modifications performed in the app so it meets the requirements outlined in sections 2 and 3. The presentation is important. At least it must include a cover page, table of contents and conclusions.
- Android Studio Project of the *CredHub* app, modified to meet the requirements outlined in sections 2 and 3.
- The app apk signed.
- A real use case scenario must be showcased and explained in detail, where the app denies communication with a rogue web server thanks to a correct configuration of *certificate pinning*.

Not following these guidelines will negatively affect the mark obtained in the assignment.