Miguel Ángel Díaz Bautista
Roberto Blanco Ancos

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

Module I Assignment: Introduction to Android APKs

Mobile Devices Security

Table of contents

1 Introduction....3
2 Development of an Android application....4
2.1 Flowchart of the application....4
2.2 Detailed description of functionalities....5
2.2.1 Title screen....5
2.2.2 Main screen (List of credentials)....5
2.2.3 “New Record” screen....5
2.2.4 “Show password / Export record” screen....6
2.2.5 “Import record” screen....6
2.3 Additional improvements....6
3 “CredHub” App Signing....7
4 Signature verification of the application *kontaktos.apk*....7
5 Inspection of digital signatures and digital certificates of the application *kontaktos.apk*....7
6 Interacting with the Activity Manager via ADB....8
7 Extracting an app via ADB....8
8 Module I Delivery....*9*
9 *Appendix – Remote repository of credentials*....9

1 Introduction

Mobile devices have evolved from basic models to today's smartphones, just as threats to mobile devices have evolved in parallel. Smartphones have a wider attack surface compared to basic phones of the past. Also, usage patterns of mobile devices have also evolved. Basic phones were used primarily for text messages and phone calls. Smartphones today are used for everything one can imagine, such as conducting banking transactions, connecting to social networks, web browsing, etc.

Most users are unaware of the risks and threats that exist on their mobile devices, which are overwhelmingly similar to those on a PC. While most people use some kind of protection on their desktops or laptops (eg antivirus software), they are oblivious to the need to protect their mobile devices, and stay unaware of the implications of certain actions. From this point of view, users are placing their trust in controls applied to software obtained from official app repositories, whether the App Store (Apple) or Play Store (Android). Some users even take the risk of resorting to third-party markets, where published apps usually undergo fewer security controls.

With this practice, we want to give an introduction to runtime analysis of some components of the Android security architecture via hands-on interaction. We especially want to emphasize those components related to security of applications and the tools that allow us to access relevant information. The main idea is to know the basic principles of the Android architecture, in order to analyze the security of applications by knowing their essential components.

2 Development of an Android application

The goal of this practice is the implementation of a password manager. The app will contain a list of credential records (*username/password* pairs) managed by a user on those services which require some sort of authentication, such as e-mail, banking applications, online commerce, etc.

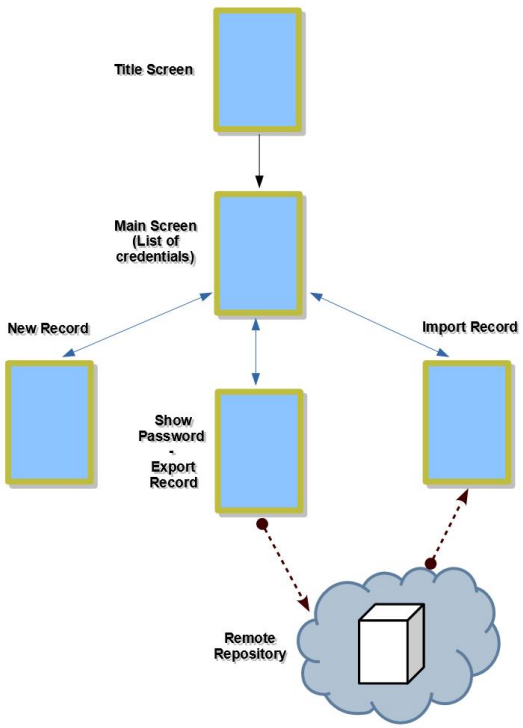
The app will maintain a persistent SQLite database of credentials on the device´s internal memory. We will also have the ability to store and extract credential records from a remote repository which will be accessed through a web service (previously provided to the students).

The new app *"CredHub"* will have the following main functionalities:

- Register a new credential record on the local database
- Display the password associated to a specific credential record
- Export a local credential record to the remote repository
- Import a remote credential record from the repository into the local database

2.1 Flowchart of the application

The flowchart of the application is as follows:



2.2 Detailed description of functionalities

This section describes the operation of each of the screens that make up the application.

2.2.1 Title screen

The display will consist of a still image, chosen by the student group, related to the subject of the application. This image is set for 3 seconds, and then the main application screen will open.

2.2.2 Main screen (List of credentials)

This screen will let us manage the whole credentials system. The following controls must be present:

- A list of all credentials stored on the local memory of the device. Only the identifier/description of each credential record will be displayed, username and password will not be shown here.
- A button leading to the "New Record" screen.
- A button leading to the "Import Record" screen.

2.2.3 "New Record" screen

Here, the app lets the user create a new set of credentials. The following elements will be required from the user:

- **Identifier/Description:** Descriptive text, indicating which service is associated to the new credentials. This field will let the user locate a given record easily on the repositories.
- **Username:** Username used to authenticate on the service.
- **Password:** Password used to authenticate on the service.

All local credentials will be stored inside a database file associated to the app on the device. Android helps us manage SQLite databases through the usage of APIs exposed by the `android.database.sqlite` package.

Once the new credential record has been created, the app must take us back to the main screen.

2.2.4 "Show password / Export record" screen

This screen will activate when the user taps the identifier/description of one of the elements from the list shown on the main screen. The controls to be displayed are as follows:

- **Identifier/Description**
- **Username**
- **"Show password" button:** By clicking this button, the corresponding password will be displayed for a fixed time period of X seconds.
- **"Export record" button:** By clicking this button, the whole set of credentials (*ID+username+password*) will be sent to the remote repository via the provided webservice (*see Appendix*).

Once the corresponding actions have taken place, the user must have the option to navigate back to the main screen.

2.2.5 "Import record" screen

This screen will automatically connect to the remote repository and extract from it a list of all credentials that may be imported (*see Appendix*).

When the user selects an element on this list, the selected record will be requested to the webservice (*see Appendix*) and the result will be included in the local database of credentials: if a local record with the same ID already exists, the existing local record will be overwritten.

When the import operation is complete, the app must take us back to the main screen.

2.3 Additional improvements

The aim of this practice is to be familiar with the basics of Android application development. That said, the following points are not compulsory but help improve the implementation and appraised:

- Modify or delete local credential records
- Generation of random passwords
- Set the webservice URL as a variable parameter by means of *sharedPreferences*
- Improving the user interface

3 "CredHub" App Signing

To carry out this section, exercise 4.1 module 1 may be used as a reference.

Create a *keystore* for the application as detailed in the relevant section.

1. Generate the key pair for your application.
2. Fill each of the required fields while creating the corrresponding certificate.

4 Signature verification of the application *kontaktos.apk*

To carry out this section, exercise 5.1 module 1 may be used as a reference.

1. Use *apktool* and mount the *kontaktos.apk* application that has been given as an attachment to this practice.
2. De-compile the application *kontaktos.apk*.
3. Describe at least 5 Manifest permissions exposed by the app and give a brief description of each permission: their use, application, etc.
4. Describe the difference between permissions required at runtime and static permissions.

5 Inspection of digital signatures and digital certificates of the application *kontaktos.apk*

To carry out this section, exercise 3.1 module 1 may be used as a reference.

1. Install the application *kontaktos.apk*.
2. Get the digital signature from the corresponding certificate of the app.
3. Get the cryptographic hashes for the app´s logo and for at least three different image files with different density of pixels. Describe the algorithm used for the respective hash encoding.
4. Describe each part of the *CERT.RSA* file of this app.
5. Give a brief description and the values obtained from these app certificate elements:

    - Serial Number
    - Validity
    - Coding algorithm used for public key
    - Size of the public key
    - Modulus
    - Signature algorithm

6 Interacting with the Activity Manager via ADB

To carry out this section, exercise 6.1 module 1 may be used as a reference.

- Get a screenshot of the partial list of applications installed on the emulator used for development, showing the package corresponding to *kontaktos.apk* application and the application created by you (*CredHub*).
- Get a list of all installed activities on *kontaktos.apk* and on *CredHub*.
- Pick an activity from *CredHub* and run it via ADB: describe the obtained results.
- Pick a service from *kontaktos.apk* and run it via ADB: please describe briefly.
- Describe how to kill both the activity and the previously released service.

7 Extracting an app via ADB

To carry out this section, exercise 7.1 module 1 may be used as a reference.

- Get a list of all permissions and creation dates for *CredHub* and *kontaktos.apk*.
- Get a list of all application resources for *CredHub* and *kontaktos.apk*, as well as their metadata.
- Get a list of all the databases for *CredHub* and *kontaktos.apk*.
- Get a description of the tables and columns in the database of *CredHub*.
- Extract all credentials stored inside the *CredHub* database file by using the "*DB Browser for SQLite*" tool.

8 Module I Delivery

- The practice will be held in groups of two.
- Practice should run on an emulator with the following characteristics:
  - Device: Nexus 4
  - Minimum SDK: Android 7.0 – API Level 24
- A memory that collects the work done on points from 2 to 7 of this document must be delivered. In addition to this, the Android project and the signed APK file corresponding to the developed app will be presented, along with a brief textual description of its functionality.
- The memory for this task must have an extension according to what is requested in each section (and it must include a cover page).
- The font used must have an appropriate size, for example 11 or 12 (depending on the font).
- Misspellings negatively influence the score.
- Each exercise must be answered by indicating its number and paragraph.
- The memory has to include screenshots and comments which help to clearly understand every concept.
- The improvements suggested on section 2.3 are not mandatory, but if implemented they will be taken into account when qualifying the practice depending on their complexity.

Failure to follow instructions negatively influences the score.

9 *Appendix – Remote repository of credentials*

A stand-alone application has been provided to the students, which acts as a web server: ***SDM_WebRepo.jar***. This program keeps the TCP port 80 open on the computer where it is running: this way, it is possible to exchange data with it by using the HTTP protocol.

The web server starts by running the following command:

```
"%JAVA_HOME%\bin\java" -jar SDM_WebRepo.jar http
```

This web server can also be configured to require authentication based on a specific username and password (*user=**sdm**; pass=**repo4droid***), and it additionally has the ability to manage secure TLS (HTTPS) connections:

- Web server activation with HTTP + Basic Authentication:

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

- Web server activation with HTTPS + Basic Authentication:

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

These protection mechanisms (authentication and TLS) will be used during the development of future practices on this subject.

The different tasks that can be requested to the web repository are the following:

- *Credentials List*: Returns a list of identifiers corresponding to those credentials currently stored on the webservice's memory.

  Notation: `String[] ListCredentials();`

- *Import Record*: Returns all values associated with a credential record, given its identifier/description.

  Notation: `String[] ImportRecord(String id);`

- *Export Record*: Stores a new credential record (*ID+username+password*) on the web repository. If a record with the same ID already exists, it will be overwritten.

  Notation: `String ExportRecord(String id, String username, String password);`

In order to establish client-side connections from the Android app towards this webservice, we will reference the library **`ksoap2-android`** *("ksoap2-android-assembly-3.6.3-jar-with-dependencies.jar")*. Students will be provided with a demo java-based application ("*SDM_WebRepo_ClientTest.java"*), where the usage of this library and connections to the remote web repository are explained.