Miguel Ángel Díaz Bautista
Roberto Blanco Ancos
Mar Giménez Aguilar

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

Module II y III: Reverse Engineering, Repackaging and Secure storage of persistent data

Index

1 Previous note: Adaptations for effective teaching during the non-attendance period

In the framework of the teaching adaptations introduced in the subject as a consequence of the closure derived from the management of the COVID-19 virus, the practical teaching of the subject has been rethought. In particular, the following modifications have been made:

1. The modules have been **adapted** to be developed individually. However, if students prefer to continue working in groups, they will be able to do so, simply by addressing the parties indicated for this purpose.
2. **Both modules are synthesized in a single task**, which will be carried out incrementally to promote continuity in learning. Thus, there will be an intermediate delivery, with a value of 0.75 points for the final grade, and a delivery towards the end of the course (in the last week of teaching) with a value of 2.25 points for the final grade. In order to promote maximum use, the final delivery may correct aspects of the intermediate.
3. The **final** delivery will be accompanied by a **brief oral defense**, **which will be done in person or remotely** as established by the regulations in force in that week.

2 Introduction

The objective of this module is to apply some techniques used by developers who want to analyze their app's security and/or by malicious developers who intend to modify that app. Someone using the techniques covered in this module usually pursues one of the following goals (which may become part of an illegal activity):

1. To add malicious behavior.
2. To bypass an intended functionality.

Furthermore, 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.

The objectives for the assignment are:

- Understanding how a particular apk is decompiled and repackaged.
- The general principle of obfuscation.
- To have a general view of how we can obfuscate our code using the android SDK - ProGuard tool, which shrinks, optimizes, and obfuscates the code by removing unused code and renaming classes, fields, and methods with semantically obscure names.
- Usage of smali and baksmali features to unpack and repack an apk.
- Traffic analysis of the data transmitted by an app, in order to find indicators of compromise.

- Use cryptographic libraries for Android, generate encryption keys, store them using Android KeyStore and use them to encrypt a database with SQLCipher.

- Increase the level of trust in connections to remote services by using HTTPS

Reverse engineering might be an illegal activity, thus it should only be performed on our own applications. The following tools will be of use in this module:

- Dex2jar: http://code.google.com/p/dex2jar/
- JD-GUI: http://java.decompiler.free.fr/?q=jdgu i
- ApkTool: https://ibotpeaches.github.io/Apktool/
- ProGuard documentation: www.proguard.sourceforge.net
- 010 Editor: http://www.sweetscape.com/
- Notepad ++: http://notepad-plus-plus.org/
- Wireshark: https://www.wireshark.org/
- SQLCipher en: https://www.zetetic.net/sqlcipher/
- Dalvik opcodes: http://pallergabor.uw.hu/androidblog/dalvik_opcodes.html

3 Modification of the "CredHub" app

Your first task is to modify the CredHub app designed in module 1. The modifications will be as follows:

- The first activity, after the title screen, will be a login screen asking for a username and password (see Figure 2.1).
- If the authentication process succeeds, the user will be taken to the main activity.
- Those tasks requiring access to the remote repository will be modified in such a way that HTTP queries will contain an "Authorization: Basic" header that includes the username and password entered during the login phase.

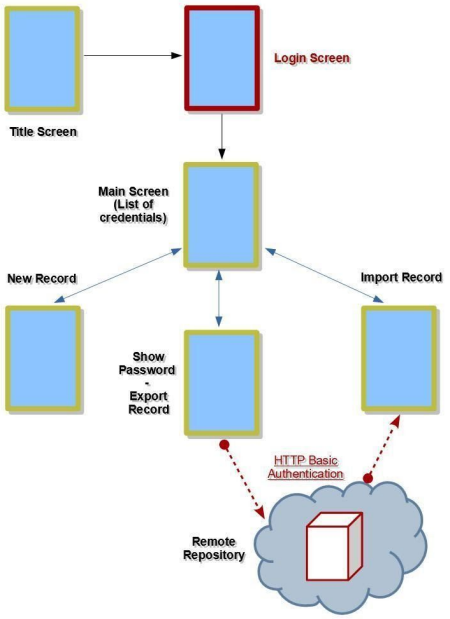

Figura 2.1. Updated workflow of the *CredHub* app.

Login credentials (username and password) will have to be previously stored in a XML resources file of the app. The password cannot be stored in plaintext, but by means of a hash function, similarly to the process explained on the exercises for this module (app uc3m_banca.apk).

We must keep in mind that, for the web server to be able to handle these new queries that include authentication headers, the server´s executable file must be launched using the following input arguments:

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

The user and password necessary for this kind of authentication are in the server client test code.They will be the same that will be necessary for the Login screen (access data).

4 Reverse engineering and repackaging

This part of the assignment will consist of:

1. Obfuscating the developed app.
2. Decompile your apk and modify it using smali code, so that you are able to login with:
    a. Mandatory: A different password from the original one.
    b. Optional in order to achieve the maximum grade: Any password.
3. Repackage the application with the changes performed on previous steps, resulting in two modified apks, one with the modifications necessary to achieve the condition 2.a and another with the ones to achieve the condition 2.b (if implemented).
4. Re-sign the modified applications and install them in the emulator.
5. Run the application in order to verify that the user name and password have been changed to values chosen by the student in case 2.a and with any value in 2.b (ifimplemented). Access any of the functionalities that require access to the remote repository in both cases and observe if its operation is as expected.

a Additional section for groups of 2 students

6. Launch a version of the app created before the repackaging process (meaning, the apk resulting from the obfuscation performed on step "4.1") on the emulator, having initiated the emulator itself through the following command:

```
emulator -tcpdump webtraffic.cap -avd <emulator_name>
```

7. After this, close the emulator and analyze the HTTP traffic generated inside the file webtraffic.cap using Wireshark: describe all the relevant information found during this analysis

5 Data protection

a Encryption of the CredHub app database

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, once ciphered, will be stored locally.
- Transparent encryption/decryption of the database using SQLCipher, using the private key once the alias has been given.

When the modified 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.

b Additional section for groups of 2 students: Communications encryption

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.

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
```

6 Module II and III delivery:

For the intermediate delivery, whose date will be announced in Aula global, the following will be required:

- The modification described in Section 3
- The modification described in Section 4, section 2.a), and the following steps (repackaging, resigning and execution).

The following must be delivered:

- Memory in PDF format that describes the steps taken to complete the tasks described in point 3 and 4, taking into account that only the modification described in section 2.a will be mandatory. Modification 4.2.b, if the student decides to implement it, can be delivered in the final delivery.
- Android Studio project with the modified CredHub application (not obfuscated) that incorporates the activity of the login screen and the HTTP authentication header and its signed and obfuscated .apk (section 4.1).
- The .apk of the CredHub application recompiled and signed with the modifications indicated in section 4.2.a.
- Text file indicating valid username and password for the necessary cases.

For the final delivery, the following will be delivered:

- The memory previously delivered, with the corrections that have been wanted to introduce, and with the explanations of the remaining points, including the optional modification 4.2.b. if implemented.
- The .apk of the CredHub application recompiled and signed with the modifications indicated in section 4.2.b, if implemented.
- Android Studio project of the modified CredHub application that meets the requirements described in section 5.
- The corresponding signed apk of the app with the requirements of section 5 and without the modifications of section 4.2.b.

In addition, if any student or group wishes to advance parts of the final delivery to the intermediate, they may do so to advance their correction.