Security Elements in Distributed Mobile Architectures

Vlad PETRE
IT&C Security Master Program
Department of Economic Informatics and Cybernetics
The Bucharest University of Economic Studies
ROMANIA
vlad@vladpetre.com

Abstract: The mobile phones have emerged in order to overcome the basic need of human beings for being in contact one with each other. In time, the functionalities of such devices have evolved. And so, the users have started to use more and more online social services like Facebook™, Twitter™ or Foursquare™, in order to keep in touch more easily. Unfortunately, the current information exchange models come with a high risk when it comes to sensitive areas like keeping the users' data private and protecting the intimacy of the users. The purpose of this paper is to make an audit of such a mobile distributed architecture – a real time location based mobile social network that aims at helping its users to remain in permanent contact with their beloved ones.

Key-Words: Android, social network, geolocation, real time, security, audit, mobile distributed architecture.

1. Introduction

In recent years, the online social networks have known a tremendous evolution and expansion. This phenomenon has been actively sustained by the fact that the human beings are constantly looking to be in close contact one with each other, as well as by the fact that the technology has evolved so much, that it is now easier than ever to build products that properly cover such communication needs [1].

The majority of the today's social networks are oriented towards the friendships between the users, towards the users' needs of sharing as much information as possible about themselves and their life, with other users (usually their friends), and towards the desire of making their location at some point in time, public.

Even though these social networks provide functionalities that fundamentally rely on private user data (e.g.: name, photos, geolocation), unfortunately the concern for protecting such private data is not the number one priority of these service providers because the benefits in terms of usability and user satisfaction are more significant than the possibility of data loss. This is related to the debate between the usability of a service versus its security [2] as many of the default enabled services can provide personal date security problems.

The purpose of this paper is to present few high risk elements that can be frequently found in such mobile solutions, in order to raise the concern on this topic. A real time location based mobile social network is audited. The proposed solution is strongly oriented towards close friendships between the users, by providing key functionalities like:

- the possibility for a user to exchange private messages with other users;
- the possibility for a user to see the current location of other users;
- the possibility for a user to share his own location with other users, in real time.

2. Background

There are mobile users that want to have a permanent connection with their close friends or relatives. There are also certain (temporary) situations in which we may be part of a social group, and wanting to have a permanent communication channel with all the group members (e.g.: a trip to the mountains).

Today, there are three main online social networks: Facebook, Twitter and Foursquare. Facebook is a social platform focused on the friendships between the users – a user being strongly encouraged to have as many friends as possible.

Twitter is a social platform focused on information spread – a user being encouraged to have as many followers as possible. And Foursquare is a social platform focused on users' locations – a user being strongly encouraged to make his geographical position known and connected with a point of interest at a certain moment in time (e.g.: a check-in at a museum, coffee shop, park, etc.).
The proposed solution to be audited in this paper comes with an extra feature that aims at better covering a need that is not yet very well exploited by the other main social networks – sharing a user's location in real time.
The proposed solution is called FriLoc and it is addressed towards users having very close friendships between themselves.
The FriLoc platform is a social network based on the real time location of the users. The platform is primarily addressed towards avid users of online social networks that also possess an Android smartphone and an active mobile Internet subscription.
The proposed solution encourages its users to interact with their friends by exchanging private messages, seeing the current location of their friends, as well as sharing their own location in the network.
A key element to understand about our users, in order to better know their needs, is the fact that the real time geolocation of a person is an element that should be treated with a maximum degree of intimacy. This means that our users completely trust their friends, as well as our network.
Taking into consideration all these elements as well as the functionalities that it provides to its users (exchange of private messages and real time location sharing), it is safe to say that the proposed solution should be an excellent support for the purpose of this paper.

3. Key Functionalities

In this section, I would like to structure in detail the key features that FriLoc provides to its users, according to interaction types:

User-User Interactions
- the user can share his own location with other users in real time;
- the user can chose who should be the users that should have access to his location;
- the user can see the location of his friends on a map;
- the user can see the current address (city, street and street-number) of his friends;
- the user can send and receive private messages with his friends.

Platform-User Interactions
- the user will be notified if a friend of him is in the same proximity area;
- the user will be notified if he receives a private message from another user.

User-Platform Interactions
- the user is able to authenticate in the network using his Facebook credentials;
- the friends of a user are exactly the same as the ones on Facebook;
- the user is able to deactivate the notifications for when receiving a private message, or when a friend of him is sharing the same proximity area.

4. Platform Architecture

In order for the platform to provide all the desired functionalities, the following components had to be configured or developed from scratch:
- a web service composed of several communication and processing modules;
- a database providing support for storing users' locations and users' messages using a queue framework;
- a Facebook application allowing secure access to the Facebook API;
- a client application that is fully compatible with the Android platform.

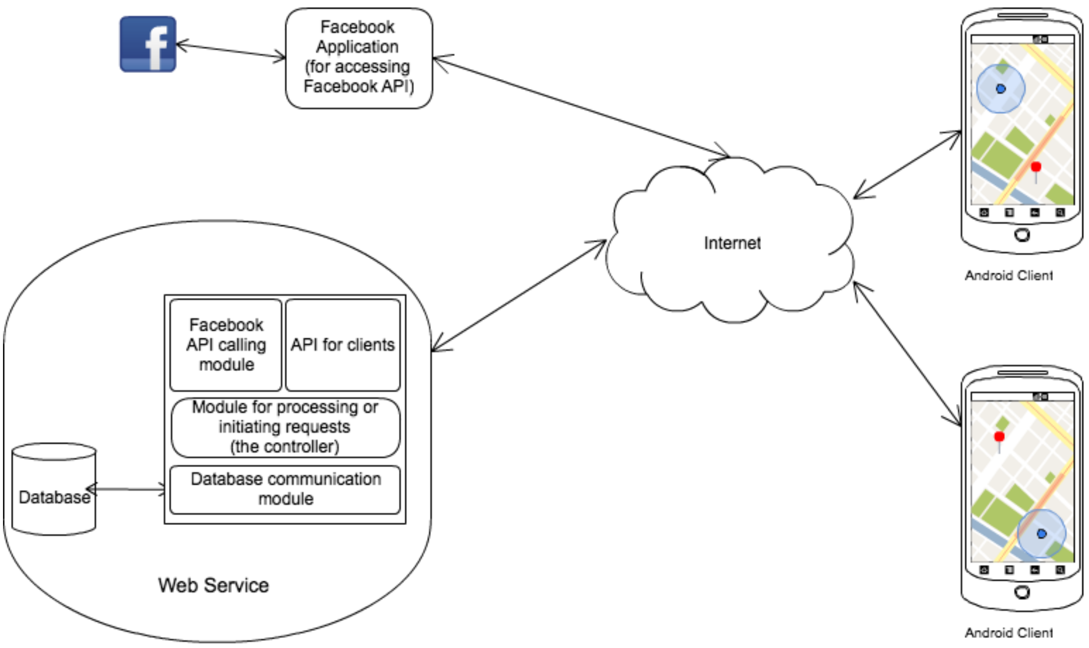

Figure 1. Architecture Diagram

Technical details about each component will be provided in the following sub-sections.

The web service is a RESTful service (Representational State Transfer), written in PHP 5.

The web service is composed of four basic modules:

- the module responsible for communicating with the database;
- the module responsible for communicating with the Facebook API;
- the module responsible for communicating with the clients;
- and the module responsible for processing the requests coming from the clients`.

The database is a relational database, with MySQL being chosen as a Database Management System. This choosing is backed up by the fact that MySQL is the most popular open source DBMS on one hand, and on the other hand, the web service being written in PHP, we have to take into account that the PHP language is has been significantly optimized for working with MySQL databases.

As an interface driver between PHP and MySQL, I have opted for MySQLi (MySQL Improved), because the API of this driver is object oriented, it offers support for transactions, and it is proved to have a higher performance than the standard one.

The authentication with the Facebook service is based on OAuth2.0 protocol. And in order to keep the data private, I have opted for the HTTPS as a secure transport protocol.

In order to make use of all the necessary features for our platform, exposed by the Facebook API, the following permissions have to be authorized by the users when authenticating in the network:

- offline_access - the permission is necessary for obtaining an *access token* from the Facebook platform over OAuth2.0 protocol that never expires;
- email - the permission is necessary in order to gain access to the user's e-mail address;
- read_friendlists - the permission is necessary in order to gain access to the user's list of friends.

The client mobile application was developed for Android 2.1 and above, and it was optimized in order to be conservative with the battery and the data consumption.

In the background there is a service that runs permanently, and that regularly

informs the web service about the user's location.
The user's location is obtained through a technique called GSM triangulation that provides an acceptable error range of around 100 meters in urban areas. [3]
The engine of the web service is in charge with processing all the requests coming from the clients, and offering a reliable response in a very small amount of time. If needed, the engine can also initiate requests to the Facebook API.
The engine is composed of a series of intelligent algorithms, for solving the most common use cases in a smart and optimal way.
As an example in this respect, we may think about the notification for when a friend is in the same proximity area as the user. The problem arises when the user has a daily routine of going to a certain location, like going to work or going to school. In this case, the user will be literally assaulted by notifications letting him know that he has friends in the same proximity area (co-workers or colleagues).
The solution for this problem consists in an algorithm that learns this habit of the user, so that it will no longer notify him of regular friends in the respective regular area.
In order to represent the data when transported over the network, I have opted for the JSON format. This solution is very popular today for transporting data from a web component to another, because it successfully manages to keep the amount of over-head to a minimum, as opposed to the older XML format.
All character strings are complaint with the ISO-8601 protocol.

5. Security Elements

In order to gain access to the Facebook API, it is mandatory to first create a Facebook Application, in order to obtain a unique application id, and an application secret. These two elements will allow the Android Facebook SDK integrated in the client, as well as the PHP Facebook SDK integrated in the web service, to communicate with the Facebook API.

The vast majority of the objects provided by the Facebook API require a valid authentication and user authorization, as a mandatory prerequisite, in order to be accessed. In order to facilitate this, a user specific access token is mandatory to be passed as a parameter, when performing the requests to the Facebook API.
An access token is specific to a certain user, and a certain Facebook application. Whenever a valid access token is provided, the associated user is considered as being successfully authenticated in the network, and the associated application is considered to be successfully authorized to perform certain actions on behalf of the user.
An access token is obtained when the user successfully authenticates with his Facebook credentials in the network, and successfully authorizes the Facebook application by granting it certain permissions. Normally, an access token expires after a certain amount of time, unless the offline permission is granted.
Because FriLoc allows its users to exclusively authenticate using their Facebook credentials, and because the standard and official Facebook SDK has been integrated into the platform in order to facilitate this, it is implicit that the underlying authentication protocol used is OAuth2.0 [8].
OAuth2.0 is a very popular federation protocol nowadays, because it simplifies the authentication/authorization processes, and offers access to protected user data to the authorized third party, without exposing the credentials of the user.
Basically, the OAuth2.0 [8] protocol allows a user that has an account on a website (the service provider) to authorize another web service (the client) to access his private data, stored in the database of the service provider, and accessible using the service provider's API. This is possible, because in reality, the users are offering access tokens to third parties, instead of directly giving them their credentials (which should always remain private!).
Besides the advantage of keeping the users' credentials private, the OAuth2.0 protocol also has the advantage of

offering limited access to the resources of the service providers. This means that the third party will be able to use as much user data as the user has granted it, upon the initial authorization.
In order to keep the web service as secure as possible over time, it is highly recommended to constantly perform routine maintenance procedures, in order to keep its components up to date:

- Apache web server;
- MySQL database;
- PHP module.

Another way of securing the web service is to make sure that all the input data is validated before it is being processed or stored in the database. This best practice significantly reduces the risk of data loss, or data leakage to bad intended users.
In order to prevent DoS and DDoS attacks against the web service, we can opt for a hosting service provider that provides a throttling mechanism, so that it controls the velocity of the requests against the web service.
In order to prevent espionage and sniffing attacks, all the communication channels of the solution use the HTTPS protocol as a secure transport protocol for data.
The client is actually an Android application. This means that the client implicitly benefits of the security mechanisms provided by the Android platform to its applications.
Each Android application runs in its own process, and has its own unique instance inside the Dalvik virtual machine. The Dalvik virtual machine runs executable files that have a *.dex* format. These files are optimized in order to use as less memory as possible.
The Dalvik virtual machine is based on the Linux kernel, and it is in charge with managing the execution threads and the memory at the hardware level.
Once installed on an Android device, an application *lives* in its own virtual memory space. This concept is called *sandboxing*:

- the Android operating system is a multi-user system based on Linux, in which each application is assigned a unique user;
- by default, the system allocates a unique Linux user identifier for each application – this unique identifier is known and used only by the operating system;
- each process has its own virtual machine, so that the code of each application is executed in an isolated way, comparing to other applications;
- by default, each application runs in its own Linux process. The operating system starts this process whenever a component of the application has to be executed. The process is stopped when the respective component ends its natural lifecycle, or the system runs out of memory.

Relying on the above security features, the Android platform implements *the principle of the least privilege*. This principle makes sure that, by default, each application has access only to the components that are necessary for it, in order to properly function.
By implementing *the principle of the least privilege*, the system is able to provide a secure environment in which an application cannot access certain system components for which it has not been granted the necessary permissions.
In order to increase the flexibility degree, additional security mechanisms have been implemented in the Android platform, in order to allow an application to share data with other applications, or to gain access to certain system services:

- it is possible for two Android applications to share the same Linux user identifier. This will allow respective applications to share files with each other. In order to conserve system resources, the applications sharing the same user identifier can run inside the same Linux process, and can therefore share the same virtual machine. But in this particular case, it is mandatory for the two applications to be signed with the same digital certificate;
- an Android application can ask the system the permission to access certain user data stored in the system

like user's contacts, short SMS messages, the SD memory card, the photo camera, the Bluetooth interface etc. All the requests initiated by an application have to be approved by the user at installation.

Because the system runs each application in a separate process, with a certain set of permissions that restrict access to other applications, an Android application cannot activate a component of another application by itself. However, the system can.
In order to activate a component from another application, the initial application has to notify the system about its *intent*, and the target component of its intent. Once the system receives the intent from the initial application, it can activate the desired component from the second application.

6. Conclusions

Given the current context in which the need for mobility and availability is more pronounced than ever, in which the smartphones have a market share that is bigger than ever, in which the internet is more accessible than ever, in which the users want to communicate as often as possible with their friends, and in which the third *country* in the world is an online social network, I strongly encourage the developers of such mobile solutions to treat the security topics of their solutions with a maximum degree of seriousness, in order to protect private user data that should normally remain confidential.

Acknowledgement

Parts of this research have been published in the Proceedings of the 7th International Conference on Security for Information Technology and Communications, SECITC 2013.

References

[1] Facebook. *Facebook Mobile: 100 Million and Growing.* https://www.facebook.com/blog/blog.php?post=297879717130 , accessed on 08.05.2013.
[2] Catalin Boja, Mihai Doinea, *Usability Vs. Security In Mobile Applications*, Proceedings of the 12th International Conference on INFORMATICS in ECONOMY (IE 2013), Education, Research & Business Technologies, Bucharest, Romania, 2013, pp. 138-142
[3] Gorumors. *Most Accessed Social Network From Mobile Phones.* http://gorumors.com/crunchies/most-accessed-social-network-from-mobile-phones/ , accessed on 08.05.2013.
[4] Daniel Plohmann, Paul Imhoff. *Localization (GPS, GSM, WLAN).* XP Course, Universitaet Bonn, 2007.
[5] Daniele Quercia, Neal Lathia, Francesco Calabrese, Giusy Di Lorenzo and Jon Crowcroft. *Recommending Social Events from Mobile Phone Location Data.* 2010.
[6] Stéphane Distinguin, Pierre-Yves Platini and Amaury de Buchet. *Social network websites: best practices from leading services.* Fabernovel Consulting, 2007.
[7] P. Mitra and K. Baid. *Targeted advertising for online social networks.* Department of Computer Science & Engineering, Indian Institute of Technology, 2009.
[8] OAuth Group, OAuth 2.0 Protocol, http://oauth.net/2/