UNIVERSITÉ CATHOLIQUE DE LOUVAIN
LOUVAIN SCHOOL OF ENGINEERING

Privacy concerns with everyday technologies

Case study of Android phones and wireless cameras

Supervisor:
Pr Gildas Avoine (EPL/ICTM)
Readers:
Xavier Carpent (EPL/ICTM)
Vianney le Clément (EPL/ICTM)

Thesis submitted in partial fulfilment
of the requirements for the degree
Master 120 in Computer Science
(option *Networking and Security*)
by **Martin Trigaux**

Louvain-la-Neuve, Belgium
Academic year 2011-2012

Contents

General introduction

- Aujourd'hui technologie partout
- L'impact sur la vie privée est souvent négligé
- A décidé de se concentrer sur deux technologies
- Le smartphone
  - Android, très à la mode auj
  - Nombre d'appareils en pleine croissance
  - Nombre de virus en croissance également
  - Contient d'en plus en plus de données personnelles
  - Qu'en est-il de la localisation d'un appareil ?
- La caméra de surveillance
  - Prévu pour nous protéger (argument souvent mit en avant)
  - Apparition de petites caméras personnelles wifi abordables
  - Est-ce que ces caméras sont vraiment sécurisées
  - Analyse en détail d'un modèle en particulier
  - Si les faiblesses trouvées sont parfois propre à ce modèle en particulier, représentatif de l'effort mit en avant pour sécurisé ces appareils
- Document structure
  - State of the art
  - android
  - camera
  - for the future

Part I

State of the art

Phone

- Téléphone portable technologie qu'on a pratiquement toujours sur soit et allumé
- Il existe plusieurs façon de localiser un utilisateur de téléphone
- Depuis le téléphone
  - Triangulation via les antennes de GSM
  - GPS présent dans quasi tous les smartphones, maintenant même dans les appareils photos
  - Wifi (cf Google, voir partie android)
  - Malware installé sur l'appareil, cas Flexispy ou Carrier IQ
- L'opérateur ou un extérieur
  - Données RAW dans les trâmes captées par les opérateurs permettent de connaitre la puissance du signal reçu, antennes...
  - SMS furtif envoyé par les autorités
  - Norme 112/911 qui oblige à pourvoir localiser n'importe qui à n'importe quel moment avec une précision relative -¿ problème du mode d'appel d'urgence sans déverouiller la carte SIM
- Scandale avec les iPhones traceurs
  - Présentation des découvertes
  - Pas rentrer dans la technique, laissée pour partie 2 section 1.2.1

RFID

- Présent dans plus en plus de produits
- Parfois toujours actif quand plus utile (sortie de magasin)
- Distance de lecture variable (*cf recherches faites pour lecture à plus longue distance*)
- Si réseau comme carte d'accès UCL était corrompu, pourrait localiser qui va où et quand (*pas fiable à 100%, rentre à plusieurs en même temps, portes pas toujours fermées*)

Wifi

- Cas Wifi UCL
  - Couvre toute la ville de LLN
  - Avec smartphone se connecte d'en plus en plus (3G encore cher)
  - Peut savoir où les étudiants se trouvent

  - Exemple étudiant prétent ne pas venir à un TP car malade, prof peut vérifier si était connecté à un wifi quelque part en ville

- Cas FON
  - Partout en Belgique depuis l'accord avec Belgacom
  - Peut connaitre les déplacements dans les villes

Part II

Android

Introduction

Structure

This part is composed of three chapters. It is advised to read them in the presented order for a better understanding as they rely on each other. However technical details can be skipped without losing the general understanding.

The **localisation using Android** chapter presents the different methods available to retrieve the location of the users. This aims to define the capabilities of an application in term of location. These methods are often unknown and considered as a black-box but when studied they may lead to privacy concerns. The anonymity and the traces left by the location requests are particularly examined in this chapter.

The **security of Android** chapter focuses on the capabilities of an application in term of security. The permission model implemented in Android is explained with its limitations and weaknesses. The different methods of an application propagation are also studied. As the previous chapter explains the capabilities of an application, this chapter demonstrates how a malicious application could end up on a user device. The malware risk has increased greatly in the previous months on Android and the security of the platform is a recurring concern. This chapter intends to clarify the risks by explaining the known possible abuses.

The combination of the two first chapter aims to develop a good understanding of the mechanisms of localisation and related security on an Android application. As an example of practical implementation of these concepts, the **DroidWatcher** mobile application has been developed. This application aims to demonstrate what an application is capable of and how any location-aware application could trace a user.

Why Android ?

Android is the operating system delivered by Google for smartphones and tablets. The choice of the Android operating system is justified by several factors.

Market share

Android is the main operating system powering smartphones with a market share of 56.1% in the first quarter of 2012[1]. As the smartphone presence increases, it is today most likely that a mobile device is powered by this operating system now and

[1] According Mobile Statistics `http://www.mobilestatistics.com/mobile-statistics/`

in a close future. The iOS system from Apple is the second most popular system with a market share of 22.9% at the same period of time.

Openness implications

The difference between these two systems is mainly in the openness policy. Android is open source while iOS uses proprietary code. This choice of openness is reflected toward the manufacturers concerning the usage of the platform (every manufacturer can port Android to its devices) but also to the developers in the management of applications. The direct opposition between the two platforms has implication on the security of the system. The open model of Android allows more abusive usages and may lead to an increased risk of malicious application propagation. The combination of high market share and security concerns makes it an interesting case study.

Not specific to Android

Although this thesis looks at the Android case, we believe our findings are relevant to other system as well. Today, every smartphone system has location capabilities similar to the one used in Android and these capabilities could be used in a malicious behaviour without the user realising it. The malicious application propagation is directly depending on the platform[2], but the risks exist and malicious applications are present on any systems that allows such abusive behaviour.

Why the localisation ?

Nowadays, the localisation privacy is often an aspect being neglected. The success of applications such as Foursquare, which intends to actively and consciously publish his current position on social networks for instance, is a clear sign of this tendency.

However, this lack of focus on privacy respect has been understood and exploited by advertisers and developers. Mobile technology brings a new dimension in the constant search for knowing more about people: the *Where am I?*. The targeted advertisements and user profiling is now common and the location is directly related to the efficiency of these techniques. If a country-based advertisement in a game application may not bother a user, the traceability possibilities are a lot more accurate and worrying. Knowing the same game application is able to accurately trace the movements of a user through time, even when he does not use the application, should be a more important concern. Demonstrating these possibilities in a transparent way is the aim behind the DroidWatcher application.

The location aspect is only studied in the capabilities of requesting the location of a device. With the exception of the cache file described in Section 1.2.1, the forensic inspection of an Android device has not been studied and could be included in future researches.

[2]The difference of an application publication on the official distribution medium between Android and iOS is studied in Section 2.3.1

Android lexicon

To help the understanding of the following chapters, some specific terms are explained below:

ROM: Android is an operating system targeting mobile devices using a Linux kernel. Literally, ROM stands for *Read Only Memory*, the part of the system where the firmware and system applications are stored but the term ROM is often used to designate a new firmware that will replace the existing one. The ROM installed by the manufacturer is usually called the *stock ROM*. CyanogenMod[3] is a well-known alternative Android ROM. Its development is driven by the community[4] and ported to a large number of devices. The installation of a ROM such as CyanogenMod does not modify deeply the system but usually allows more configuration, correct some bugs or extends the possibilities of the device.

Root: on most devices, the owner of an Android device has only a limited access to the system. It can install new applications but not see or alter the content of the device system file where the application is stored. Gaining a root access on a device is a procedure to access all the capabilities of the system and be able to modify it. The procedure for root access varies from device to device since root access protection is left to manufacturers choice and can require complex procedure[5]. Gaining root access is considered sufficient to void the warranty of a device with many manufacturers. Having a root access is required to install an alternative ROM. To realise the experiments described in Section 1.5, a root access was necessary to be able to monitor every aspect of the system.

[3] CyanogenMod `http://www.cyanogenmod.com/`

[4] Unlike Android which is open source but developed by Google only

[5] The website ready2root gathers procedures to gain root access on a large list of devices `http://ready2root.com/`

Chapter 1

Localisation using Android

Introduction

The localisation of the mobile device is a key system service of Android. Many applications use this service, as a feature of the application.It allows, for example, to directly show the part of the map where the user is on Google Maps, to display advertisements for nearby shops, to automatically select the relevant area for the weather forecast, to give statistical information regarding the using of an application in a particular region.

The details of the localisation methods used by the Android system are often unknown or unclear. There exists several techniques to locate a device: using the GPS chip usually present in smartphones or using connection information (cell phone tower or wireless access points). This chapter explains how these techniques work and why they are subject to privacy concerns.

Several facts about the management of the localisation have been used in the official declarations or documentations. To verify these assumptions, several experiments have been carried out using an Android 2.3 smartphone. These experiments allow us to have a better understanding of the localisation system that mostly works as a black box.

1.1 GPS

1.1.1 GPS signal

The GPS, for Global Positioning System, is a technology based on satellite trilateration [4]. GPS satellites are navigating around the Earth in a way to maximise the number of visible satellites anywhere at any time. There is currently 31 working satellites in orbit. The location-aware device is equipped with a GPS receiver chip. This receiver retrieves broadcasted messages from the reachable satellites. The messages contain :

- the time the message was transmitted
- precise orbital information (the ephemeris)
- the general system health and rough orbits of all GPS satellites (the almanac)

Trilateration is used based on the received messages as shown in Figure 1.1. If, theoretically, three satellites are enough, at least four are required to avoid clock synchronisation errors (at the speed of light, even a small clock error can lead to huge distance gap).

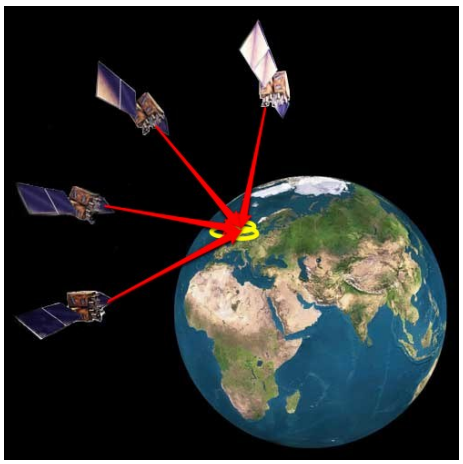

Figure 1.1: Signals from multiple satellites are required to calculate a position
Copyright PocketGPSWorld.com

The accuracy of this method depends on the surrounding of the user. In clear sight, the GPS is accurate to a few meters but it will degrade if the receiver is surrounded by high buildings or inside a dense forest.

The time to first fix (TTFF) depends on the state of the GPS. In a cold start scenario[1], the GPS needs to retrieve the full almanac. This is done in at least 12.5 minutes [5]. To improve the TTFF, embedded GPS often uses assisted-GPS technology (aGPS) by acquiring almanac and ephemeris data over a fast network connection when available.

1.1.2 Degradation

The GPS system was invented by the U.S. Department of Defence as a military project. As the technology became available to civilians, it was intentionally degraded. To do so, the satellites used a technology called Selective Availability intended to introduce errors on a GPS signal to decrease its accuracy. The authorised users (military) could compute the errors and correct them using special receivers and daily cryptographical keys. The Selective Availability was implemented for national security reasons.

In 2000, Bill Clinton ordered to discontinue the usage of the Selective Availability feature on the GPS satellites, allowing each receiver to perceive the most accurate signal possible. In 2007, the new generation of GPS system called *GPS III* was announced as not capable of producing Selective Availability [3].

[1] Device is in factory state or the GPS data are not relevant (several months old or inaccurate)

1.2 Wireless access point

Each wireless access point has a unique identifier[2]. When the wireless option of the phone is turned on, the device can retrieve the surrounding access points. Assuming the geographical coordinates of all the access points are known, it is possible to estimate the location of the user by trilateration.

The advantage of this method is that for an accuracy of about 100 meters, the localisation is faster than using GPS, it works indoors, consumes less battery power than a GPS receiver chip and only one access point is enough to locate a device.

1.2.1 Cache database

To locate a device using the network resources (as opposed to the GPS resources), the system needs an access to a database mapping the geographical coordinates of the requested access points. SkyHook, Apple and Google are three companies well known for using such databases.

SkyHook was one of the first to create a database to locate wireless access points and develop a SDK to query the location of a user. The information is collected by war-driving[3] in North America, Western Europe and some Asian countries [7].

Companies have quickly understood the value of this information and the economical interest of having its own database as a betterment for location aware applications. While Apple was, at first, using SkyHook, it has now developed its own database system. Google is also independent and has created its own database.

As they collect data to improve the accuracy of their location services, these companies recently have been subject to criticism. Users wondered about the usage of this database and how it could pose a threat to the privacy of users. The privacy aspect of the localisation section is analysed in Section 1.4.

1.2.2 Methods of data collection

In the current study of the Android system, Google solution is taken as an illustration. The location server is constructed based on two factors:

- Google Cars
- Crowd-sourcing

The Google Cars are mainly used to take pictures to illustrate the service Google Street View. In addition to that, the cars are also war-driving. Having a GPS module on the car and driving almost all over the world, it was a good opportunity to constitute a very accurate database.

As most Android devices are equipped with a GPS receiver, collecting via crowd-sourcing is also possible. When an Android device uses the Google database to

[2]BSSID for Basic Service Set IDentification, a unique 48 bits address
[3]Car equipped with a GPS, wireless and cell tower receiver collecting data in the streets

request a location, data are also transmitted to Google servers. This way, the database of wireless access points and cell towers is always up to date[4].

1.2.3 Location cache files

Previous cell tower and access point locations are stored in an unencrypted system files. This allows to locate the user quickly and the device is still able to use the network resource, even when not connected to the Internet. Each entry in the cache file is linked with a timestamp representing the date of the retrieval as seen in Figure 1.2.

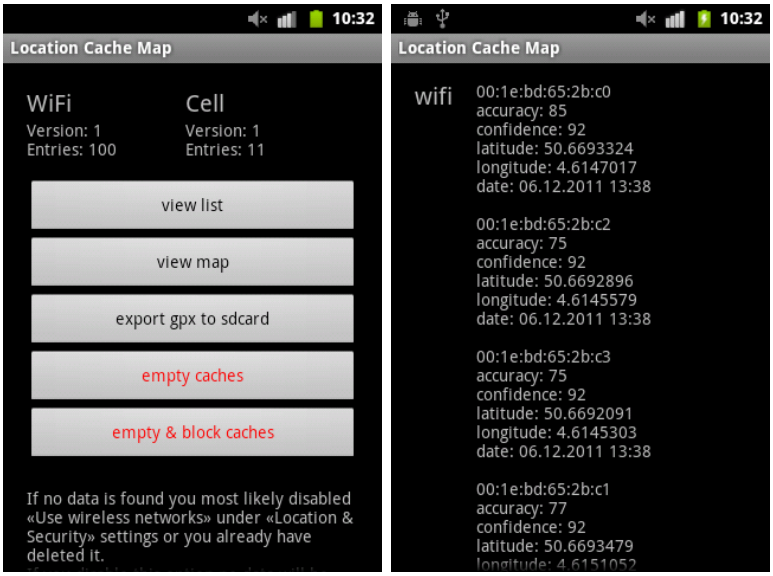

Figure 1.2: Captures from the application Location Cache by Remy Demy

The recent criticism about privacy concerns is mainly based on the existence of these cache files. If the iOS devices used to have unlimited cache size (fixed in iOS 4.3.3), on an Android device, only the 50 last cells and 200 access points have been observed as stored in the cache files. However, in the course of this research, the observations made based on the analysis of the data collected from several devices have shown that this size is enough to contain locations older than one month. A forensic analysis would allow to retrieve the device location at a given time if a location request was made. Such requests can be run in background by any application having the correct permissions. The DroidWatcher application relays on this fact to monitor the location of the user in background.

To show the ease of retrieving such information, a python script has been developed [1] to parse the content of these two files and produce a GPS trace file in GPX format[5] representing the approximate movement of the device. However, a root access to the phone is required to access these files[6]. This could prevent

[4]Francisco Kattan, Feb 2010, `http://franciscokattan.com/2010/02/06/dynamic-cell-id-clever-way-to-block-google-but-will-it-backfire/`

[5]GPX is a XML file format listing geographical coordinates with timestamp and allowing to reproduce the movement of a device

[6]Most of the time, granting a root access requires a system manipulation and voids the manufacturer warranty.

malicious applications retrieving these data. This cache folder is only created when the *Use wireless networks* option is enabled in the Android settings. However, this option is often required by common applications such as Google Maps which may lead to a large percentage of devices having this option enabled.

1.3 Cell tower

Similar to the method used with wireless access points, a cell tower can be identified in a unique way. A GSM cell tower is characterised by two factors: a Location Area Code regrouping tens or hundreds of cell towers and a Cell ID identifying a cell tower inside a location area. It is the combination of these two factors that allows a device to identify a unique cell tower. A cellphone can then be located using trilateration based on the surrounding cell towers.

The data collection method and cache management system are very similar to the ones analysed in Section 1.2 and will not be discussed here. The advantage of using cell tower location over the wireless is the fact it allows to locate a device with an approximate accuracy of 1km almost everywhere.

1.4 Privacy concerns

1.4.1 Google Cars

In May 2010, Google admitted to German authorities having collected more than what it was supposed to. In addition to access point's unique identifier, it had "been mistakenly collecting samples of payload data from open networks". These data chunks could include parts of web surf, email, text...[7]. In reaction, the data collected was asked to be deleted and the CNIL (independent French administrative authority) fined Google with €100.000[8].

1.4.2 _nomap

Some users considered the collected data by the Google Cars and Android devices as private. In November 2011, in reaction to criticism, Google created a way to opt out recording of its access point. The proposition of Google is to end the ESSID of the wireless access point with `_nomap`. The next time it is scanned by a Google Cars or an Android device, the access point is removed from the database. Google hopes than over time, the `_nomap` string will be adopted by other location providers [2].

This proposition was received with much of scepticism and did not satisfy the pro-privacy groups. The main complain was the need of an action from the user to explicitly opt out its access point while people wants a way to explicitly opt in instead. Many people that are concerned by privacy issues do not have enough technical knowledge to modify the wireless network name. Furthermore, if this

[7] TechEYE, May 2012, `http://news.techeye.net/security/google-admits-it-sniffed-out-peoples-data`

[8] BBC UK, Mar 2011, http://www.bbc.co.uk/news/technology-12809076

string is not universally adopted by other companies such as Apple or SkyHook, conflicting systems can be imagined, preventing a concerned user to fully opt out its access points from commercial databases.

1.4.3 Research of Samy Kamkar

To reply to privacy concerns, Google ensured "The location information sent to Google servers when users opt in to location services on Android is anonymised and stored in the aggregate and is not tied or traceable to a specific user" [8]. The security researcher Samy Kamkar has also looked at the location requested.

He succeeded to decrypt the request made to Google servers and realised that it contains a unique identifier [6]. The identifier is unique to the cell phone and present in every request. If this string does not directly reveal the identity of the phone owner, it is however possible to tied the string to a specific user and then trace him. He affirms there is no proof the location is anonymised due to the presence of this identifier.

1.5 Personal researches

Several facts about the storage of information and privacy have been announced. To verify these facts, further investigations have been carried in this thesis. The applications used to realise the analyses are present in the appendix.

To realise the experiments a rooted smartphone using Android 2.3 was used. As most of the location information, including the cache databases files are located in restricted part of the system, the rooting was necessary to explore the whole content of the phone. The root process and implications are discussed in the introduction.

1.5.1 Experiment 1: Database suppression

Goal

When the option *Use wireless networks* in the system settings is disabled, Google ensured the cache files are deleted. The cache files have been located long ago but the asked question was to know if it was the only place that stored this location information.

During this experiment, the content of the device is inspected before and after opting out the option *Use wireless networks* and the two versions are compared to detect the modified or deleted files.

Methodology

1. Make a dump of the internal memory
2. Disable the *Use wireless networks* option
3. Make a dump of the internal memory
4. Compare the two dumps

To facilitate the experiment realisation the two scripts `androdump.py` and `compare_dump.py` have been developed and are available on the CD-ROM. The dump is done using the script `androdump.py` which uses the Android Debug Bridge (adb) program connecting the device to a computer. The comparison is done using the script `compare_dump.py` which uses a hash function on each file present in the dump to detect a modification.

Result

The goal of the comparison was to ensure that only the folder containing the databases is altered and the information is not stored somewhere else. The analysis reveals that, with the exception of some irrelevant system files (battery state...) modified, the database files only are updated. The cache data of Google Maps application has been also deleted. This confirms the assumption concerning the deletion of location data.

1.5.2 Experiment 2: Impact of location requests

Goal

When a location is requested on an Android device, the system sends an encrypted request to the Google's location servers. The Google's servers reply to the request with the location of surrounding GSM cell towers and access points.

The goal of the experiment was to detect the impact of a location request. It is known that the surrounding wireless access points and cell towers information is stored in the cache file but other location related information could be stored somewhere else. This experiment inspects the content of the system before and after a localisation request is made.

Methodology

1. Make a dump of the internal memory
2. Request the current location
3. Make a dump of the internal memory
4. Compare the two dumps

The Android application `LocateMe` has been developed to create a simple location request using the network provider (including GSM cell towers and wifi access points).

Result

The analysis reveals that, with the exception of irrelevant system files, only the database cache files have been modified.

1.5.3 Experiment 3: Correlation between size and cache of requests

Goal

As the request is encrypted, its content is unknown. What is known and confirmed in the previous experiments is the fact that the cache files are updated and some wifi and cell towers information are added to these files when a request is made. To guess the content of a location request, it has been tried to determine patterns in the requests form to correlate a request with the content of the cache files.

Methodology

1. Start in *blank state*[9]
2. Start monitoring the traffic
3. Activate the wireless
4. Request the current location
5. Stop monitoring the traffic once a location received

	Time	Source	Destination	Protocol	Length	Info
33	19.885665	192.168.1.12	209.85.148.99	TLSv1	146	Client Hello
34	19.922270	209.85.148.99	192.168.1.12	TCP	66	https > 55496 [ACK] Seq=1 Ack=81 Win=5696 Len=
36	20.022875	209.85.148.99	192.168.1.12	TLSv1	1484	Server Hello
37	20.023392	192.168.1.12	209.85.148.99	TCP	66	55496 > https [ACK] Seq=81 Ack=1419 Win=8736 L
38	20.024815	209.85.148.99	192.168.1.12	TLSv1	373	Certificate, Server Hello Done
39	20.025152	192.168.1.12	209.85.148.99	TCP	66	55496 > https [ACK] Seq=81 Ack=1726 Win=11572
40	20.123988	192.168.1.12	209.85.148.99	TLSv1	252	Client Key Exchange, Change Cipher Spec, Encry
41	20.164498	209.85.148.99	192.168.1.12	TLSv1	113	Change Cipher Spec, Encrypted Handshake Messag
42	20.165073	192.168.1.12	209.85.148.99	TCP	66	55496 > https [ACK] Seq=267 Ack=1773 Win=1157
44	20.220910	192.168.1.12	192.168.1.1	DNS	86	Standard query PTR 99.148.85.209.in-addr.arpa
45	20.273673	192.168.1.1	192.168.1.12	DNS	125	Standard query response PTR fra07s07-in-f99.1e
46	20.290005	192.168.1.12	209.85.148.99	TLSv1	269	Application Data
47	20.293773	192.168.1.12	209.85.148.99	TLSv1	916	Application Data
48	20.347040	209.85.148.99	192.168.1.12	TCP	66	https > 55496 [ACK] Seq=1773 Ack=1320 Win=953
49	20.375062	209.85.148.99	192.168.1.12	TLSv1	598	Application Data, Application Data
50	20.375513	192.168.1.12	209.85.148.99	TCP	66	55496 > https [ACK] Seq=1320 Ack=2305 Win=144
64	24.644455	192.168.1.12	209.85.148.99	TLSv1	269	Application Data
65	24.645577	192.168.1.12	209.85.148.99	TLSv1	646	Application Data
66	24.698070	209.85.148.99	192.168.1.12	TCP	66	https > 55496 [ACK] Seq=2305 Ack=2103 Win=129
67	24.725247	209.85.148.99	192.168.1.12	TLSv1	695	Application Data, Application Data
68	24.725713	192.168.1.12	209.85.148.99	TCP	66	55496 > https [ACK] Seq=2103 Ack=2934 Win=172

Figure 1.3: Example of tcpdump capture while a location request displayed in Wireshark

The network monitoring has been made with the tool *tcpdump* installed on the Android device.

Result

From the collected trace, the size of the transmitted data was compared with the number of cells and access points added to the cache files. After observation and repeating the experiment, the following observations were made. Each location requests is composed of two communications with Google's servers. In each communication, a first packet of 270 bytes is sends followed by another of variable

[9] *Blank state* : wireless turned off, empty location cache, location permission turned off, no process requiring the location such as Google Maps running.

size.

S. cell	S. wifi	Req 1	Req 2	C. cell	C. wifi
6 (1)	4	271+644/654	271+879/817	2 (1)	4
1	3	271+428/619	271+576/713	1	3
?	?	269+916/598	269+646/695	1	2

Table 1.1: Example of location requests and effect on the cache files

Table 1.1 shows data collected during the experiments realised. **S. cell** represents the number of surrounding valid GSM cell towers at the collect time. 6(1) means there is six surround GSM cell towers detected but only one is valid[10]. **S. wifi** represents the number of surrounding wireless access points at the collect time. **Req 1** represents the first communication and **Req 2** the second (271 + 644/654 means two packets of 271 and 644 bytes are uploaded and 654 bytes are downloaded from Google's servers). **C. cell** represents the number of cell towers and **C. wifi** the number of wireless access points in the cache files after the location request (numbers between parenthesis represents the number of valid entries in the cache file).

Figure 1.3 shows an example of collected trace confirming the derived pattern. The packets number 46 and 64 contain 269 bytes and are the initiating packets of a request for cell towers and wireless access points. The packets number 47 and 65 are the surrounding cell towers and access points uploaded to Google servers. The packets number 49 and 67 are the reply from Google's servers containing the coordinates of the known cell towers and access points. These observations do not allow to validate the suppositions concerning the content of the packet but the patterns are coherent to the model explained before.

1.5.4 Experiment 4: Unique identifier

Goal

In Section 1.4.3, Samy Kamkar observed a unique identifier was used in the requests made to Google servers. The presence of this identifier could compromise the privacy of the user as it would allow Google to trace a location requests to a certain user. The purpose of this experiment was to verify this fact.

Methodology

Personal researches showed that this string is contained inside a file `gls.platform.key` next to the cache database. When the option *Use wireless networks* in the Android settings is disabled, the content of the cache location folder (containing the wifi and cell cache files as well as this identifier) is emptied. When the option is enabled, new cache files and unique identifier are create. The content of the identifier file `gls.platform.key` is different to the previous value every time the option is toggled.

[10]Some cell towers are detected but displayed as using a Cell ID and LAC of -1, these are considered as invalid and ignored. These invalid cells have often a very weak signal strength.

Conclusions

The traceability possibility are limited due to this constrain. If it is relatively easy to change this value, we can however imagine that very few users are aware of the existence of this value and will apply this manipulation regularly.

Chapter 2

Security of Android

Introduction

As the number of smartphones is in constant raise, the level of concerns about the security of the system increases. Paradoxically, the users tend to store more and more personal data on their smartphones and are not aware of the security issues of such devices. Malwares have been discovered on the official applications store and antivirus softwares for Android are now available. Android runs on top of a Linux kernel which is yet reputed to be virus free.

The aim of this chapter is to explain in detail the current security mechanisms used to protect the users against malicious applications. Using the presented information, a user should be able to reduce his infection risk by adopting simple security principles. The focus of this chapter is the application capabilities and propagation. Different security threads are examined and the risks associated are evaluated. The different procedures from the publication to the installation of an application on a device are particularly examined.

The forensic aspect to retrieve information from a device without the owner consent have not been analysed here.

These clarifications are essentials to understand the limits and possibilities for the developed *DroidWatcher* (see Chapter 3) application to be effective.

2.1 Permissions

For an application to run on the Android operating system and access to critical resources, it should have explicitly been allowed to do so. For a set of defined tasks, a permission should be enabled. These tasks are, for example, accessing the current location of the user, update the address book, use the Internet, write to the SD card... At the installation of an application, the permissions necessary are mentioned.

The permission system is designed to control the usage of internal methods and resources of Android. Without a permission, an application can not access to certain resources or methods in the Android system.

2.1.1 Technical details

The full list of permissions with a brief description is available on the Android documentation[1] These permissions are defined in the configuration file `AndroidManifest.xml` present in every application. Without the correct permission, an application throws an exception when the method accessing the forbidden resource is launched.

Listing 2.1: Example of permission violation log

```
E/AndroidRuntime( 1274): FATAL EXCEPTION: main
E/AndroidRuntime( 1274): java.lang.RuntimeException: Unable
    to start activity ComponentInfo{com.example.gpstest/com.
    example.gpstest.MainActivity}: java.lang.
    SecurityException: Provider gps requires
    ACCESS_FINE_LOCATION permission
...
E/AndroidRuntime( 1274): Caused by: java.lang.
    SecurityException: Provider gps requires
    ACCESS_FINE_LOCATION permission
...
```

In Listing 2.1, is shown the Android debugger trace of an application requesting the location of the device using the GPS location provider without having requested the `ACCESS_FINE_LOCATION` permission. If the error is not caught properly, the execution of the application is interrupted and the users receives a notification of the crash of the application.

The permission processed is conceived to control the access to an information and not a phone characteristic. For example, the `ACCESS_COARSE_LOCATION` permission is not limited to the usage of the high level `LocationManager` methods but is also required for an application to retrieve the surrounding cell towers information (as these towers have a unique identifier, this lower level information could also be used to locate the user[2]).

2.1.2 Weaknesses

The way the permission system is implemented does not fully prevent malicious behaviours. The permission description is unclear and can include different purposes. For example, the permission `READ_PHONE_STATE` is required for many actions. It allows an application to be notified when a phone call is processed or when the device is locked, it also gives information about the phone unique identifier and SIM id. This permission is often used to suspend services or simply track a device using the unique identifier. The problem is that, in case of a phone call, it also provides the access to methods allowing to retrieve the caller phone number. This is an information leakage that could have been avoided.

Also, it is unclear when and why an application requires a permission at the installation process. Many free applications display advertisements to fund their

[1] Available at `https://developer.android.com/reference/android/Manifest.permission.html`

[2] This method is used in the DroidWatcher application to estimate the location even when no network connectivity is available

development. This kind of applications requires the permission to access the Internet to download the advertisement content. A malicious gaming application could justify the need for the two permissions `INTERNET` and `WRITE_EXTERNAL_STORAGE` (access the micro-SD card of the device) for advertising and score storing. Using these permissions, it could upload the full content of the SD card (which may contains personal information from the other applications) to a server. Only a deep analysis such as network monitoring can detect malicious behaviours of an application.

Finally, if a user disagrees with the need of a suspicious permission, it has no other choice than not installing the application. There is no possibility to partially accept the permissions. Due to this restriction, if they want to use the application, we can assume than most users will accept, whatever the asked permissions are.

2.2 Installation of applications

Unlike iOS where the App Store is the only permitted source of applications[3], the Android operating system proposes several ways to install an application.

2.2.1 Play Store

By default[4], the Android Play Store (formerly named Android Market before its merging with Google Music) is the only source of applications. Once a Google account is associated to the user's phone, it can use the application Google Play Store which lists the available applications and install them quickly. Figure 2.1 shows an example of the interface of the application.

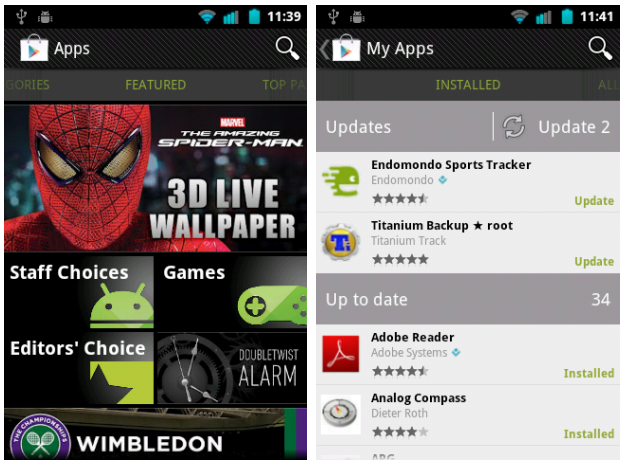

Figure 2.1: Google Play Store interface

[3]Alternative markets and applications distributions exist on iOS but they require jail-breaking which is not allowed by Apple

[4]The Play Store is available only on official Android devices approved by Google. The Android operating system is open source which allows the port on many devices but the Android Play Store application is closed sources and compatible with only the official Android devices.

The Android Play Store has several features that can be handy for the average user:

- Warning when an update is available
- Control by Google against malicious applications
- User comments and review
- Paid system with Google Checkout

On the website `https://play.google.com/store`, the content of the Android Play Store is available from a browser. An important feature of this website is the possibility, once logged with the associated Google account, to select applications to install. The next time the Android device is connected to the Internet, it will automatically download and install the selected applications without any user interaction required. A simple notice is displayed on the phone once the application is installed.

2.2.2 Other sources

By default, the possibility to install applications from other sources than the Play Store is disabled. Changing this setting is proposed when a user is trying to install a software from another source for the first time.

.apk file

The *.apk* extension is the convention for installable applications on the Android operating system in the same way as *.deb* or *.rpm* are on Debian and Fedora operating systems.

A user trying to open such files on his device launches the installation process in the same way as if he was using the Android Play Store. The required permissions are displayed and ask for the approval of the user. Figure 2.2 shows the permission screen when a user tries to install the application DroidWatcher. This is the same screen while using either the Google Play Store or installing an *.apk* file.

An apk file is produced after the compilation of a program and is often proposed on small projects or for beta versions. Once an application is installed on the system, the apk file is stored on the system.

Alternative marketplace

In the same way as the Android Play Store, it exists several alternative market places. Alternative market places are a way to download apk files from a centralised interface. It is seen as an alternative to the Google Play Store with the advantages of a centralised distribution medium (paid system, users reviews, moderation...). Manufacturers or service providers sometimes sell smartphones with their own marketplace instead of the Google Play Store.

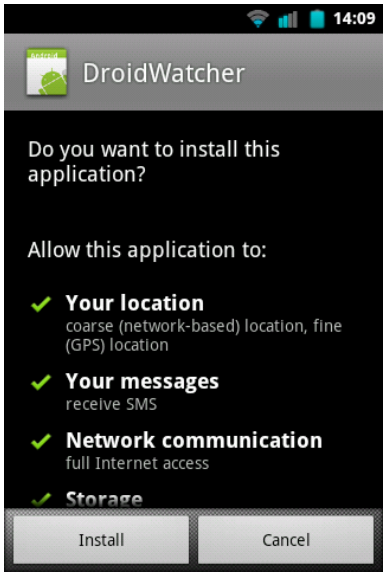

Figure 2.2: Permissions required to install DroidWatcher

Debug mode

If the debug mode of an android device is turned on (done in the configuration settings of the device), interaction between a computer and the device is possible. Using the official Android Debug Bridge toolkit[5], an application can be installed in a few seconds from a computer without any notifications or user interactions on the device connected to a computer (connection done typically using a USB cable).

2.3 Attack schemes

As the multiple installation procedures make the propagation of application easier, it also allows abuses and permit the propagation of malwares. Figure 2.3 presents the different possibilities for a malicious applications to propagate and be installed on a device. The attack types are described in Section 2.4.1.

2.3.1 Play Store publication policy

In comparison to the Apple iOS system, Google has adopted an open policy of application publication. This strategy choice makes the Android system an easier target in term of malware propagation on the official applications distribution platform.

To distribute an application on the Android Play Store, the registered developer can upload his application on Google Play servers and make it available in a few hours[6]. There is no control before the publication of an application. The upload of a malicious application would be detected only after its publication which can lead to infected users. In comparison, when submitting an iOS application, Apple will

[5]Documentation `https://developer.android.com/tools/help/adb.html`

[6]Official Distribution Control guidelines `https://developer.android.com/distribute/googleplay/about/distribution.html`

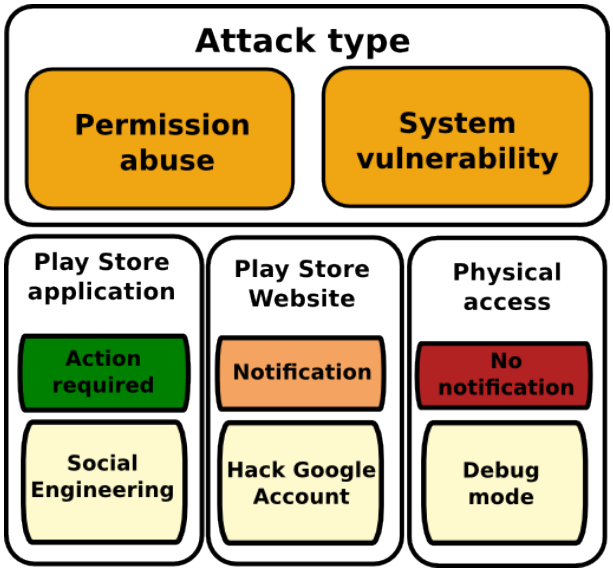

Figure 2.3: Malicious applications propagation possibilities

review the application which can takes several days.

To become a registered developer, the owner of a Google account has only to pay a 25$ fee and no other information than a name and a phone number. In comparison, to become a registered Apple developer, a minimum of 99$/year fee[7] to subscribe to the developer program is required and more personal information such as credit card information for identity verification.

This difference in the verification process and ease to create a developer profile between Google and Apple may lead to the presence of more malicious applications on the Android platform than on iOS. The presence of an application on the Android Play Store is then, by no means, a guarantee of safety and a user should check carefully before installing any application.

As for the Play Store, alternative market places are as secured as the control of their owners over the applications acceptance process. The *Amazon AppStore*[8] developed by Amazon.com Inc. has an approval process to publish applications similar to Apple regarding iOS applications[9].

2.3.2 Play Store website

In the case of an attacker gaining access to the Google account of an Android user, it could remotely install any application. This would allow the attacker to do almost any kind of actions the device is capable of. It would be possible to monitor the activity of the user or remotely command the phone. A simple notification is

[7] According to Apple Developer Programs `https://developer.apple.com/programs/`

[8] Available in the US only at `http://www.amazon.com/appstore`

[9] Approval Process and Content Guidelines `https://developer.amazon.com/help/faq.html#Approval`

displayed at the application installation that can be unnoticed or not understood as an infection sign by inexperienced users.

2.3.3 Social engineering

As the developer policy of Google is open by nature, the description of an application published on the Play Store may not describe the real behaviour of an application. Free games or widgets are often attractive and are an ideal target for a malicious application writer that can use social engineering. Pretending another purpose than the actual behaviour of an application leads to its installation willingly from the user.

This attack scheme is relevant for applications on the Play Store or installed using any other sources. However, there is frequently a confusion from the users supposing the Play Store is more secure than it actually is. This may lead to reduce the suspicion of the users and makes the attack more effective.

Also, websites have appear on the web proposing applications that are non-free on the Play Store. These applications can be instead a malware or have been manipulated to inject malicious code in the original application. Therefore, applications downloaded on such websites should be considered as are the warez websites on the Windows operating system: highly risky in term of malware propagation.

2.3.4 Physical access

If a malicious user has a physical access to a device, it can install an application from a computer using the *adb* utility in debug mode. The micro-USB connectivity is an European standard recommendation for smartphone connectivity. It is then easy to connect any smartphone to a computer with this connectivity. If the debug mode is enabled[10], there is no need to activate the phone which makes the presence of screen lock ineffective.

When a malicious application is installed using this method, no notification is displayed, the only detection possibility is the presence of the application in the installed applications list.

It is recommended to disable the debug mode when not required and use a screen lock to prevent modifying the phone settings.

2.4 Malwares

2.4.1 General malware types

There is two main types of Android malwares: abuse of permission or use of security flaws.

The first kind of malware takes advantage of the lack of suspicions from the users and simply ask for permissions allowing a malicious behaviour. This is usually the case for applications sending text messages to overtaxed number or stealing

[10]The debug mode is required for any interaction between a computer and a smartphone

contact information from the address book. This kind of malware tends to be timeless and works as long as the users do not inspect attentively the permission screen whatever the operating system version he is running. As some "honest" applications have a large range of features (that the user may never use), it is common to see such applications asking for many permissions (for example, the official Facebook application requires 19 different permissions[11]). The reasons of the asked permissions is usually not mentioned by the application makers[12].

The usage of security flaws is possible due to the slow update process. The manufacturers tends to provide only a limited number of version updates if any[13]. A device older than a year is usually not maintained anymore. The only solution for users owning such devices is to install alternative ROMs such as CyanogenMod that provides a longer support for a large range of devices. When a security flaw is discovered and a patch published, only a very small percentage of users benefits from the patch through an update in the months following the discovery. Malware writers can then wrote programs taking advantage of that flaw.

2.4.2 The DroidDream malware

In spring 2011, a malware named *DroidDream* has widely spread across the Android devices. The particularity of this malware was that he used the official Android Play Store (called Android Market at that time). Referring to Figure 2.3, it is classified as using social engineering to exploit a system vulnerability.

The attackers created several developers accounts and malicious applications (above 50 different applications were detected) on the Play Store. The applications used social engineering by taking the name of popular applications and using modified versions of the application to trick the users into downloading them. The malware used exploits effective until the version 2.2 (99% of the devices at that time[14]) to break the sandboxing mechanism, root the device and install other applications preventing the removal. The malware has been called DroidDream as it was set up to run between 11pm and 8am to contact the master server. Due to its ability to install other applications, the Kaspersky Lab's analysts suppose it could have been monetised in the future to be used as a botnet (spam sending, distributed denial of services...).

In reaction to the discovery of this malware, Google activated the *kill switch* which deleted the malware from the user devices remotely. It was the first known example of widely spread command-and-control malware on mobile devices. Researchers estimated the number of infected users between 50,000 and 200,000 de-

[11] Discovered through personal researches by decompiling the downloaded application from the Android Play Store in July 2012

[12] Counterexample: Firefox browser created a page to explain the reason each permission is used `http://mzl.la/FirefoxPermissions`

[13] Computerworld has computed the percentage of Android phones upgraded to Froyo (released in May 2010) by each manufacturer within 2010 `http://blogs.computerworld.com/17649/android_upgrades`

[14] Data collected based on the connections to the Android Play Store, source `https://developer.android.com/about/dashboards/index.html`

vices[15]. Variants of this malware called DroidDream Lite have been detected a few months later.

2.4.3 Protection

Observing the large increase of malware applications on the Android platform[16], users and developers wonder about the need of antivirus software. As the antivirus for desktop computers works, these antivirus usually work using a malware database basis. Such softwares would be efficient on antivirus using flaws and derived in several applications such as the DroidDream malware did. However, on the second kind of malicious applications, the efficiency of the antivirus is mitigated as it is very easy and quick to develop applications abusing from the granted privileges.

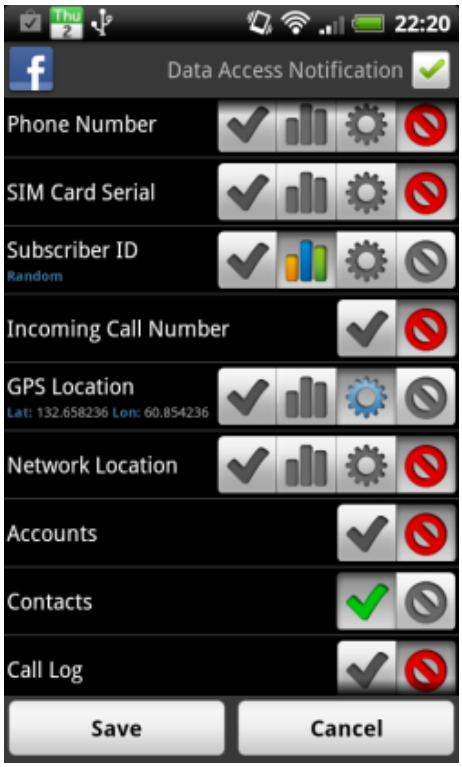

Figure 2.4: PDroid on the Facebook application

The PDroid application[17] takes another approach than the antivirus softwares. Instead of detecting the known malicious applications, it allows the users to redefine the granted permissions and revoking them at wish. Another possibility instead of disallowing the access to certain information is to define a fixed or random value (eg: in the case of geographical coordinates). For each application, a notification can be launched at the time the resource granted by a permission is used. This feature can

[15] According to the number of time the applications have been downloaded in total

[16] Between 2011 and 2012, the number of Android malware families has increased from 10 to 37 according to F-Secure `http://www.zdnet.com/blog/security/android-malware-families-nearly-quadruple-from-2011-to-2012/12171`

[17] Available on the xda-developers forum at `http://forum.xda-developers.com/showthread.php?t=1357056`

be useful to detect abuse of permissions. However, as the PDroid application works as a intermediate layer between the operating system and the other applications, the application need the root privileges and the user has to apply a patch on the ROM files. These requirements are most of the time not possible on manufactured phones with closed sources ROM and are reserved to users with good computer knowledge. Although it is not applicable to most Android users, PDroid is a possibility of big improvement on the permission model and we can hope a similar model to be adopted in future versions of Android. Figure 2.4 is an example of usage of the PDroid application capabilities on the Facebook application.

Chapter 3

DroidWatcher

3.1 Presentation

DroidWatcher is an application developed in the context of the current thesis. It aims at demonstrating the concrete accessibility of localisation services. By exploiting the official capabilities[1] of the Android operating system, a system monitoring actively the device movement has been developed. In the context of this thesis the application takes only acceptable actions. But of course more invasive actions could be taken by malicious developers.

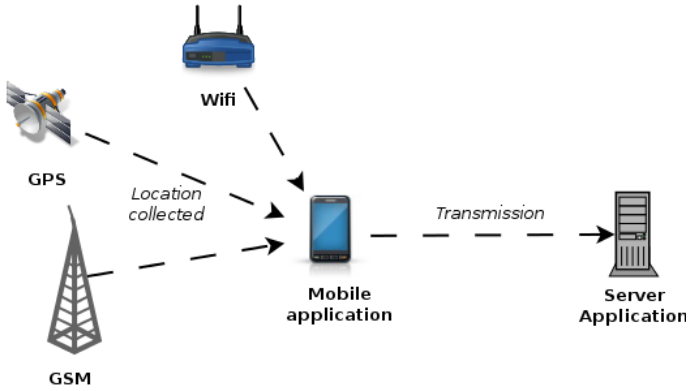

Figure 3.1: DroidWatcher system architecture

DroidWatcher is made of two major components as represented in Figure 3.1:

- A mobile application running on a smartphone:
  the application essentially collects location data on a regular basis.
- A server-based application:
  the application centralises the data collected by the smartphones as soon as those smartphones have an internet access.

This chapter presents more specifically the mobile application. Installation guide and user manual on the complete DroidWatcher solution, on the mobile and server side, can be found in Appendix A

[1] A known bug has however been exploited for the remote GPS activation feature as explained in Section 3.2.5

3.2 Mobile application execution

3.2.1 Application structure

Once the device is connected to the Internet, it synchronises itself to a remote server to upload the last recorded coordinates.

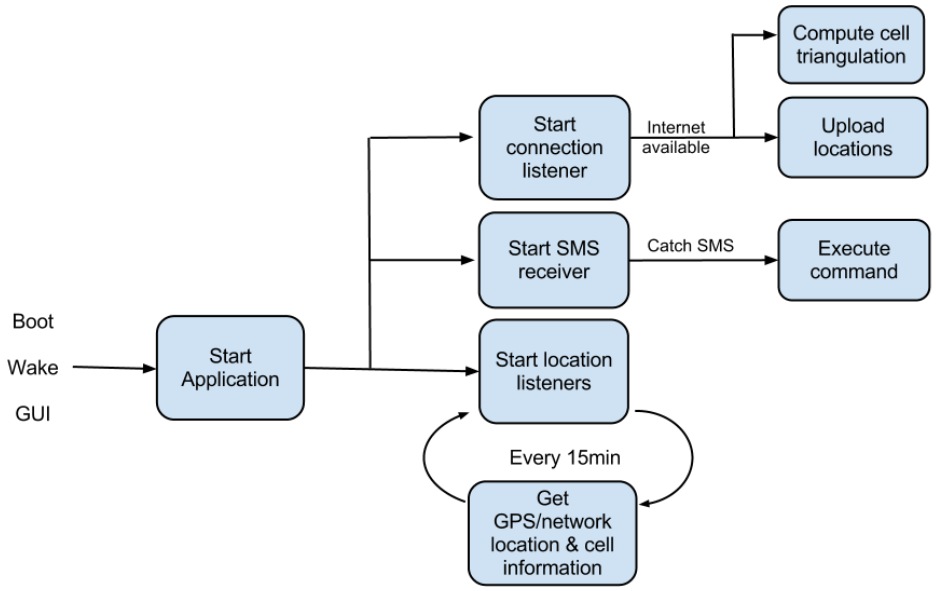

Figure 3.2: DroidWatcher execution process

Figure 3.2 shows the execution of the mobile application. The application starts at phone boot, when unlocked or when the interface is used[2]. When starting, the application has three main components:

- Periodic recording of the location and cell information
- SMS commands management
- Internet requests and synchronisation

3.2.2 Location recording

At the installation, the mobile application requests the localisation permissions[3]. Using these permissions, every specified time interval (15min by default), DroidWatcher records the location of the device. The application works in background and keeps recording locations when the user does not use its smartphone. Only the most accurate location is kept in an interval of 15min. Both GPS and network[4] resources are monitored using the official provided methods in Android. If a new

[2]Except with Android 4.0 or above where the interface should be launched at least once, see Section 3.4.2 for technical information

[3]See Section 2.1 for details about the permission model

[4]The network resource is defined as the usage of wireless and cell tower access points as described in Section 1.2 and 1.3

location is not available (the system keeps in cache the last recorded location), it will be ignored. The locations received are stored in a file on the SD card.

The surrounding cell data are also collected for future location. The cell towers identifiers and signal strengths are recorded at the same frequency as the GPS and network locations. The geographical coordinates of the collected GSM cell towers are retrieved once the phone is connected to the Internet and computed afterwards by using triangulation.

3.2.3 Internet actions

Periodically and when the wireless is enable on the smartphone, the application verifies if the device is connected to the Internet. When it is the case, the application takes two actions:

- Retrieve the geographical coordinates of the collected GSM cell towers and compute the previous locations using triangulation
- Synchronise the collected locations to the remote server

The coordinates of the GSM towers are collected using an unofficial Google database available at `http://www.google.com/glm/mmap`. This database has been selected as it is one of the most complete compared to the other free alternatives. However, due to its unofficial state, it is possible the data will not be available in the future.

Using the geographical coordinates of the cell towers, the application is able to compute by triangulation of the previous location of the device. The method developed to triangulate a device is explained in Section 3.3.

3.2.4 SMS management

The mobile application provides a SMS-based command interface. These commands allow to configure the application (frequency of retrieval, server url...) and take direct actions related to the location (enable GPS, returns current location...). The application intercepts the received SMS before the main SMS application[5]. If a predefined code is detected, it will discard the text message and will take an action accordingly. Otherwise, the SMS is ignored and will be delivered to the main SMS application.

The complete list of SMS commands is available at Appendix A.1.3.

3.2.5 GPS activation

The GPS of a device can be remotely activated using an SMS command. This feature is possible due to a bug discovered in the power control widget[6]. Even if the security flaw has been revealed in April 2010 and a patch released in April 2011,

[5]The application is set to use the maximal priority in the chain of events. However if another application has also set the maximal priority, it may retrieve the SMS before the DroidWatcher application.

[6]Issue 7890 `https://code.google.com/p/android/issues/detail?id=7890`

the flaw has been observed as still exploitable on devices running Android 2.3.

This is the only part of the software where a bug is exploited in this application instead of using the official systems capabilities. However it is important for an user to understand such flows exists and, even when corrected, can affect a large part of devices (in August 2012, Android 2.3 and below represented 81% of the Android market share).

3.3 Cell tower triangulation

When a device needs to record the current location but is not capable to access the Internet, the program records the surrounding cell towers information. The described triangulation algorithm is applied once the coordinates of the collected cell towers are retrieved, the next time the smartphone is connect to the Internet.

3.3.1 Scenarios

To triangulate a device using the cell towers, the following scenarios have been considered:

1. One GSM tower is within range
2. Two GSM towers are within range
3. More than two GSM towers are within range

The case where no GSM tower is within range is not considered as the location can not be estimated and the algorithm is not used. To evaluate the location, a *centre point* is decided and a *confidence range* is computed. The device is estimated to be located within the area covered by the circle drawn using the centre point and the confidence as radius.

It has been decided not to differentiate the eventuality of more than three different cell towers as it will greatly increase the complexity of the algorithm without increasing the accuracy of the method proportionally, as intended by the limitations explained in Section 3.4.3.

3.3.2 Determine the position

Depending of theses three scenarios, the according centres will be decided as:

1. The location of the only GSM tower within range
2. Along the median between the two GSM towers
3. *The intersection between the two closest towers that is on the side of the third closest tower* Reformer

If more than three GSM towers are within range, the closest towers are determined as the ones with the strongest signal strength.

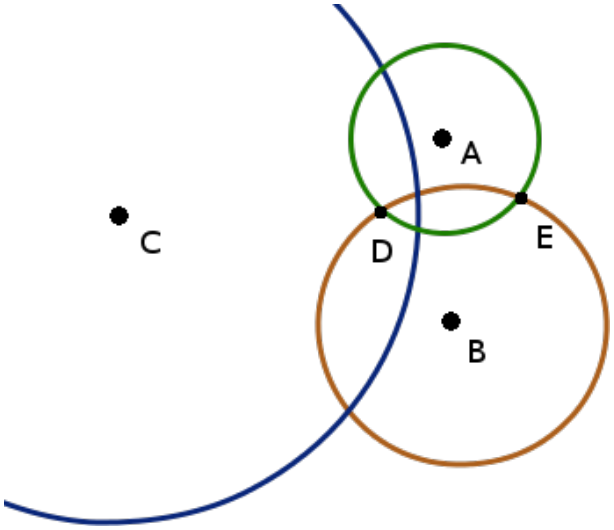

Figure 3.3: Triangulation scenario with 3 GSM towers

The third scenario is illustrated in Figure 3.3. The tower A and B are considered as the closest to the device as received with the highest signal strength, the tower C is the third closest to the phone. For each tower, a circle of a radius size related to the signal strength is drawn. Two points D and E are computed as the intersection between the circles of centre A and B. The point D is determined as the best centre point as it is the one closest to the tower C and is then evaluated as the location of the device.

In the three considered cases, the confidence range is determined as

1. The conversion of the signal strength to meters
2. The distance between the two intersections
3. The half of the distance between the two intersections

Comment justifier l'utilisation de la distance ?

3.3.3 Signal strength conversion

If a device receive a GSM signal at a determined strength, it is possible to estimate its distance from the cell tower. A circle is drawn around the GSM tower representing every position the device could be. The strongest the signal is, the closest is the device to the tower.

The signal strength collected by the Android device is expressed in *ASU*, for Arbitrary Strength Unit, which is a value between 0 and 31 derived from the signal strength usually computed in decibel[7]. 0 ASU representing the lowest signal and 31 the strongest. The way to express the conversion is made using the following formula :

[7]For the GSM network, $dBm = 2\text{Ö}ASU - 113$

$$(-\sqrt{asu} + 6) * 900$$

The weighting of the factors has been decided based on personal observations using the author's smartphone.

3.4 Technical challenges

To develop this application, several constrains were experienced that limited the effect of the application.

3.4.1 Automatic idle

Once going in idle state (once the device is not used by the user for a certain amount of time), the operating system will limit the possibility of the system to save battery. Some applications will be paused in there running process. The effectiveness of the localisation process done by DroidWatcher is affected by this idle purpose.

This is the case, for example, of the GPS that needs to constantly update of the position. The GPS will sometime stop monitoring the position of the user if it can not get a fix[8] on the location of the user. This effect is independent of the application but the direct consequence of the system behaviour for battery saving. This issue is often a complain related to the tracking application (eg: sport monitoring application). A solution is avoid the phone to going to idle state by keeping it in awake state. However, this solution is not used in DroidWatcher as it would have greatly compromised the battery usage of the phone and consequently the effectiveness of the application in monitoring the location the longest and most discrete way as possible.

3.4.2 Android 4.0

As the author of this thesis owns only a device with the Android version 2.3. At the time of development, end 2011, the fourth version[9] of Android was just released and very few devices were capable of running it. Consequently the testing has been done mainly on devices running the second version of the operating system.

An unexpected change introduced in the 4.0 version[10] of Android is the way a device manage the start of an application in background. DroidWatcher has been conceived to be started when a device is booting or waked from idle. This feature participated in the aim to be fully discrete and that the application was not noticeable without analysis. With Android 4.0, an application can no longer start during the boot or after having been woken up if the interface has not been launched a first time [**boot-restrictions**].

[8] See Section 1.1 for the information needed to get a GPS fix

[9] The third version of the operating system was limited to tablet devices and not phones limiting greatly the propagation of this version of the system.

[10] The change was already present since the version 3.1 for tablets but reflected to smartphones only since the version 4.0

To fix this problem, an interface screen has been developed. This screen allows the user to see location information and basic configuration.

This change is certainly an improvement in the security of a device as the need for a graphical interface will strongly reduce the possibility of malicious *invisible* application to run. However malicious applications often use a fake interface (weather forecast, game...) to hide the malicious behaviour of the software and this protection will therefore not affect these applications.

3.4.3 Cell tower triangulation

To compute the location of the user, a triangulation algorithm has been developed. This algorithm is however known as imprecise for several reason. To compute the location, the algorithm uses the signal strength of captured GSM cell towers nearby. The signal strength is a very fluctuating variable. At a same distance to a cell tower, the signal varies if the device is inside or outside a building or if monitored by two different devices with different GSM receiver. The main imprecision comes from the fact that all cell tower do not emit signal at the same signal strength (rural areas are usually covered with less cell towers emitting using higher signal strengths.

As efficient computation of the signal strength would have required long monitoring and observation on a large number of devices and areas. The computation and weighting of the variables has been done based on personal observations. This is known as imprecise but achieves the purpose to be able to record a reasonable approximation of the location at any time when GSM connectivity is available.

Part III

Camera

Chapter 4

Camera

- Camera analogiques
- Camera sur le web
  - Retrouve dans Google
  - Non protégées
- Wireless camera

Chapter 5

Case analysis: DLink camera

- Présentation de la caméra
  - Caméra entrée de gamme
  - Cas particulier mais globalement valable
- Code du source
  - Disponible suite à décision de justice
  - Problèmes pour le récuperer
- Securité
  - Protocoles utilisés
  - Compte invité
    - Pas mentionné
    - Pas supprimable
  - Fichier de log
    - Bug
    - Accessible par tous
  -

Appendix A

DroidWatcher guides

A.1 Mobile application

A.1.1 Installation

The installation of the mobile application is similar to the installation of most Android application.

1. Retrieve the file DroiDWatcher.apk from the CDROM[1] to the Android device
2. Open the file with the Android installer
3. Accpet the requested permissions
4. Open the interface DroidWatcher in the menu pannel to launch the background application

A.1.2 Interface

For configuration ease and to solve the restriction appearing on Android 4.0 as explained in Section 3.4.2, a configuration interface has been created. This interface allows to see the last location computed and the date of the last synchronisation to the remote server. The option are also given to specify a specific data collection URL and choose if the phone will or not reply to SMS commands.

A.1.3 SMS commands

The application can be controlled through SMS commands sent to the phones running the application. The messages are intercepted before arriving to the message application. If the message contains a pre-defined code, the phone will execute an action in consequence.

- The messages are not case sensitive.
- The match should be exact (no extra character).
- The application does not record the content of messages, the messages not containing the code will not be affected.

[1]The file can also be downloaded at `https://gitorious.org/martin-trigaux-thesis/droidwatcher/blobs/raw/HEAD/mobile/DroidWatcher.apk`

`BIGBRO` : starting code for a command.

- `LOCME` : reply with the last recorded location
- `GPSON` : turn the GPS on
- `WIFION` : turn the wireless on
- `SETSERVER[new_server_url]` : set the url of the server, default `http://watcher.dotzero.me/collect`

Examples of correct messages:

- BIGBROGPSON
- bigbroSetServerhttp://watcher.dotzer.me/collect
- Ping

Examples of incorrect messages

- BIGBRO GPSON
- Ping!

To easily test if the application is running, the message `PING` can be send, the targeted cell phone replies with message containing `PONG`.

Turning on the GPS is done by exploiting a bug in some Android roms. It was reported as working on v2 Android ROM and CyonengMod 7.

A.2 FAQ

A.2.1 What data is collected by the application ?

- Estimated location and time of the recording
- Google username
- IMSI (International Mobile Subscriber Identity)
- Phone number (if written in the SIM card, usually not)

The Google username is collected to easily differentiate the users while the IMSI and phone number are to ensure the uniqueness. Note that the IMSI and phone number do not require any permission and that any application can collect it.

A.2.2 When run the application ?

The application start at the phone boots and when the user unlock its phone. Except if using Android 4.0 or above, killing the process will only stop it until the next time the phone is unlocked. Uninstalling the application `DroidWatcher` will fully remove it.

A.2.3 What is stored on the phone ?

The last collected cell towers and last locations are collected in the file `.log.obj` at the root of the SD card. You can remove this file safely.

A.2.4 Who is able to see the recorded location ?

To ensure privacy, only the owner of the server is able to see the collected location.

A.3 Installation of the web application

To watch the collected information, the DroidWatcher collecting website can be installed on your own web server. The server use the python framework Django 1.3[2]. The following steps explain the deployment of the application on a Debian Lenny server running Apache and mod-wsgi. The full configuration and security of the apache server is considered as out of the scoop of these explanations.

1. Download the latest version of Django
   ```
   $ wget http://www.djangoproject.com/download/1.3.1/tarball/ -O
   django.tar.gz
   ```

2. Extract and install
   ```
   $ tar -xzvf django.tar.gz
   $ cd Django-1.3.1
   $ sudo python setup.py install
   ```

3. Extract and deploy the DroidWatcher Django application from the DroidWatcher package
   ```
   $ tar -xzvf watcher.tar.gz
   $ sudo mv watcher /var/www/watcher
   ```

4. Change the ownership to the apache user
   ```
   $ sudo chown -R www-data:www-data /var/www/watcher
   ```

5. Update the apache configuration file (probably `/etc/apache2/sites-enabled/000-default`) and add

   ```
   <VirtualHost *:80>
               ServerName SERVERURL
               Alias /static/ /var/www/watcher/static/
               <Directory /var/www/watcher/static>
               Order deny,allow
               Allow from all
               </Directory>
               WSGIScriptAlias / /var/www/watcher/apache/django.wsgi
   </VirtualHost>
   ```

6. Update eventually the django setting in `watcher/settings.py` files if you want to configure your email or have changed the location of the application folder.

[2] Available at `https://www.djangoproject.com/`

7. Generate the database. In the application folder, execute
   `$ python manage.py syncdb`
   and choose an admin password.

8. Restart the apache module
   `$ sudo service apache2 restart`

9. Access the received location by going to `http://SERVERURL/admin` to log in and them access to the recorded location at `http://SERVERURL/`

Bibliography

[1] Magnus Eriksson. *Android location dump*. URL: https://github.com/packetlss/android-locdump.

[2] Google. *Greater choice for wireless access point owners*. URL: http://googleblog.blogspot.com/2011/11/greater-choice-for-wireless-access.html.

[3] GPS.gov. *Selective Availability*. 2012. URL: http://www.gps.gov/systems/gps/modernization/sa.

[4] Darren Griffin. *How does the Global Positioning System work ?* 2011. URL: http://www.pocketgpsworld.com/howgpsworks.php.

[5] US Coast Guard. *Navigation Center's NAVSTAR GPS User Equipment Introduction*. 1996. URL: http://www.navcen.uscg.gov/pubs/gps/gpsuser/gpsuser.pdf.

[6] Declan McCullagh. *Android data tied to users? Some say yes*. 2011. URL: http://news.cnet.com/8301-31921_3-20056657-281.html.

[7] *SkyHook Coverage Area*. URL: http://www.skyhookwireless.com/location-technology/coverage.php.

[8] *Testimony of Alan Davidson, director of public policy at Google*. URL: https://docs.google.com/viewer?a=v&pid=explorer&chrome=true&srcid=0BwxyRPFduTN2NmI2NGVjMWUtZDg0NC00NGI5LWJlYTctNmI4MGQ2YmIzYzUz.