How are we tracked in our everyday life ?

Martin Trigaux

July 21, 2012

Contents

Introduction

- Aujourd'hui technologie partout
- L'impact sur la vie privée est souvent négligé
- A décider de se concentrer sur deux technologies
- Le smartphone
  - Android, très à la mode auj
  - Nombre d'appareils en 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

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

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

- Pourquoi s'intéresser à Android ?
  - Aujourd'hui OS majoritaire sur smartphone
  - Vente smartphones >ventes pc aux USA
  - Entends d'en plus en plus des problèmes de virus
  - Confie énormément d'info à un smartphone
  - Fait moins attention à ce qu'on fait sur un smartphone qu'un pc
- La localisation est un aspect souvent négligé
  - Gens pensent *quel est le problème si mon jeu sait où je me trouve quand j'y joue ?*
  - Si arrive à tracer en temps réel les déplacements de l'utilisateur, plus inquiétant.
  - Les gens ne se rendent pas compte ce dont est capable une application.
  - A créé une application pour montrer cela, DroidWatcher
- choisi de se concentrer sur l'aspect applications, pas étudié l'aspect forensic
- court lexique (root, ROM...)

Chapter 1

Localization using Android

Introduction

The localization of the user is a key function under Android. Many applications use this functionality, as a feature of the application or as a service for the developer. 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 gives statistical information of the country usage of an application...

The details on the location methods used by the Android system are often unclear. This chapter explains how the Android system manages the location of a device and how this has been subject to privacy concerns. The efficiency and limits of the DroidWatcher application presented in Chapter 3 are dependent of the location techniques detailed in this chapter. As also been verified through experiences several facts concerning the location data storage.

1.1 Available techniques

Depending of the state of the phone, several techniques can be used to locate an Android device.

1.1.1 GPS

The GPS, for Global Positioning System is a technology based on satellites trilateration[3]. GPS satellites are navigating around the earth in a way to maximize 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 contains :

- 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 is required to avoid clock desynchronization errors (at the speed of light, even an small clock error can lead to huge error).

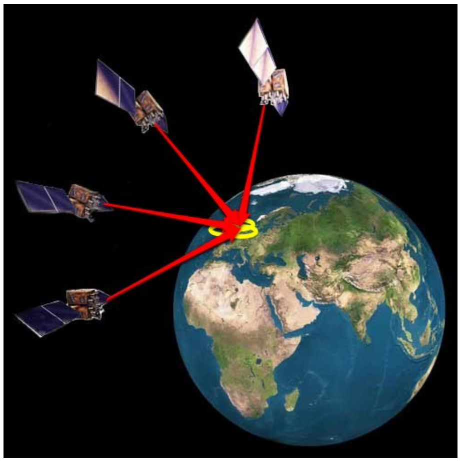


Figure 1.1: Signals from multiple satellites are required to calculate a position
Copyright PocketGPSWorld.com

The accuracy of this method depends of 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[4]. To improve the TTFF, embedded GPS often use assisted-GPS technology (aGPS) by acquiring almanac and ephemeris data over a fast network connection when available.

1.1.2 Wireless access point

Each wireless access point has a unique identifier[2]. If the wireless 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 have an estimation of the location of the user by trilateration.

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

[2] BSSID for Basic Service Set IDentification, a unique 48 bit address

The advantage of this method is that for an accuracy of about 100 meters, the localization is faster than using GPS, consumes less battery power than a GPS receiver chip and does not have other geographical requirements than being surrounded by at least one access point.

The collect method used by Google is explained below.

1.1.3 Cell tower

Similar as for the method used with wireless access points, a cell tower as a unique identifier. On the GSM network, a cell tower is characterized by a Location Area Code and a Cell-ID. A cellphone can them be located using trilateration based on the surrounding cell towers.

In the DroidWatcher application (see chapter 3), the application implements its own trilateration algorithm for offline computation.

1.2 Access points and cells databases

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

SkyHook was one of the first to create a database to locate wireless access points and developed an 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[6].

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 have been subject to criticism recently. Users wondered about the usage of this database and how it could hurt the privacy of users[4].

1.2.1 Collect method

In the case of Google, the location server is constructed based on two factors:

[3] Car equipped with a GPS, wireless and cell tower receiver collecting data in the streets

[4] In may 2011 a database containing the location of the last visited access points and cell tower has been discovered inside iPhones. Similar database is also present in Android phones.

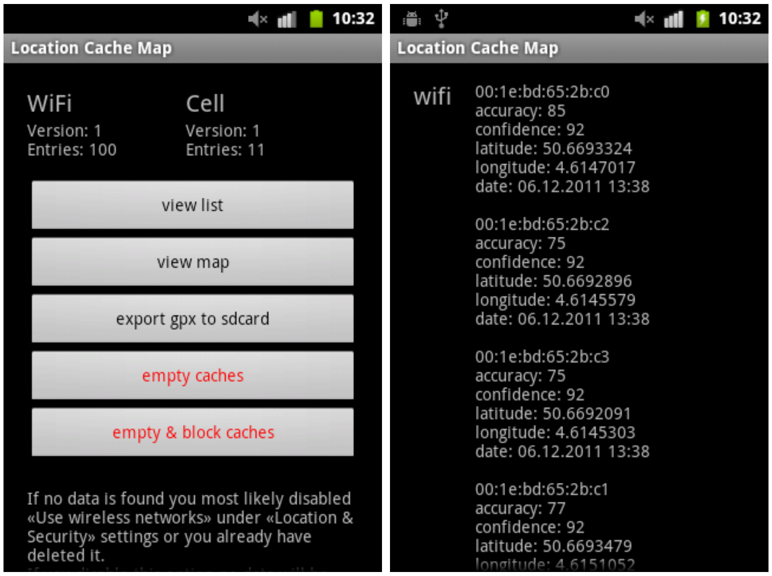
- Google Cars
- Crowd-sourcing

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 request a location, data are also transmitted to Google servers. This way, the database of wireless access points and cell tower is always up to date[5].

1.2.2 Location cache files

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



WiFi
Version: 1
Entries: 100

Cell
Version: 1
Entries: 11

If no data is found you most likely disabled «Use wireless networks» under «Location & Security» settings or you already have deleted it.

wifi

00:1e:bd:65:2b:c0
accuracy: 85
confidence: 92
latitude: 50.6693324
longitude: 4.6147017
date: 06.12.2011 13:38

00:1e:bd:65:2b:c2
accuracy: 75
confidence: 92
latitude: 50.6692896
longitude: 4.6145579
date: 06.12.2011 13:38

00:1e:bd:65:2b:c3
accuracy: 75
confidence: 92
latitude: 50.6692091
longitude: 4.6145303
date: 06.12.2011 13:38

00:1e:bd:65:2b:c1
accuracy: 77
confidence: 92
latitude: 50.6693479
longitude: 4.6151052

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

The recent criticisms about privacy concerns were mainly based on the existence of these cache files. If the iOS devices used to have unlimited cache size (fixed in

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

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, observations made on 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 permission. The DroidWatcher application relay on this fact to monitor the location of the user in background.

To show the ease of retrieving such information, a python script as been developed[1] to parse the content of these two files and output a GPS trace file in GPX format representing the approximate movement of the device. A root access to the phone is however required to access these files[6]. This could prevent malicious applications to retrieve these data. Also this cache folder is only composed 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.

1.2.3 Personal researches

Several facts about the storage of information and privacy have been announced. To verify these facts, a set of personal researches have been made. The applications used to realize the analyses are present in the appendix.

Database suppression

When the option *Use wireless networks* is disabled, Google ensured the cache files are deleted. To ensure the information is deleted from the system when the option is opt out, the following experiment was done:

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

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 reveal that, with the exception of some irrelevant system files (battery state...) modified, the database files and also Google Maps cache have been deleted.

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

Analysis of location requests

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.

To understand the content of the requests, two analyses have been made. The first one aimed to ensure the location informations are stored only in the two cache folders. The second one is an analysis of the volume of transmitted data correlated to the number of entries added in cache.

To ensure that the location information is only stored inside the cache files, the following experiment was done :

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

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

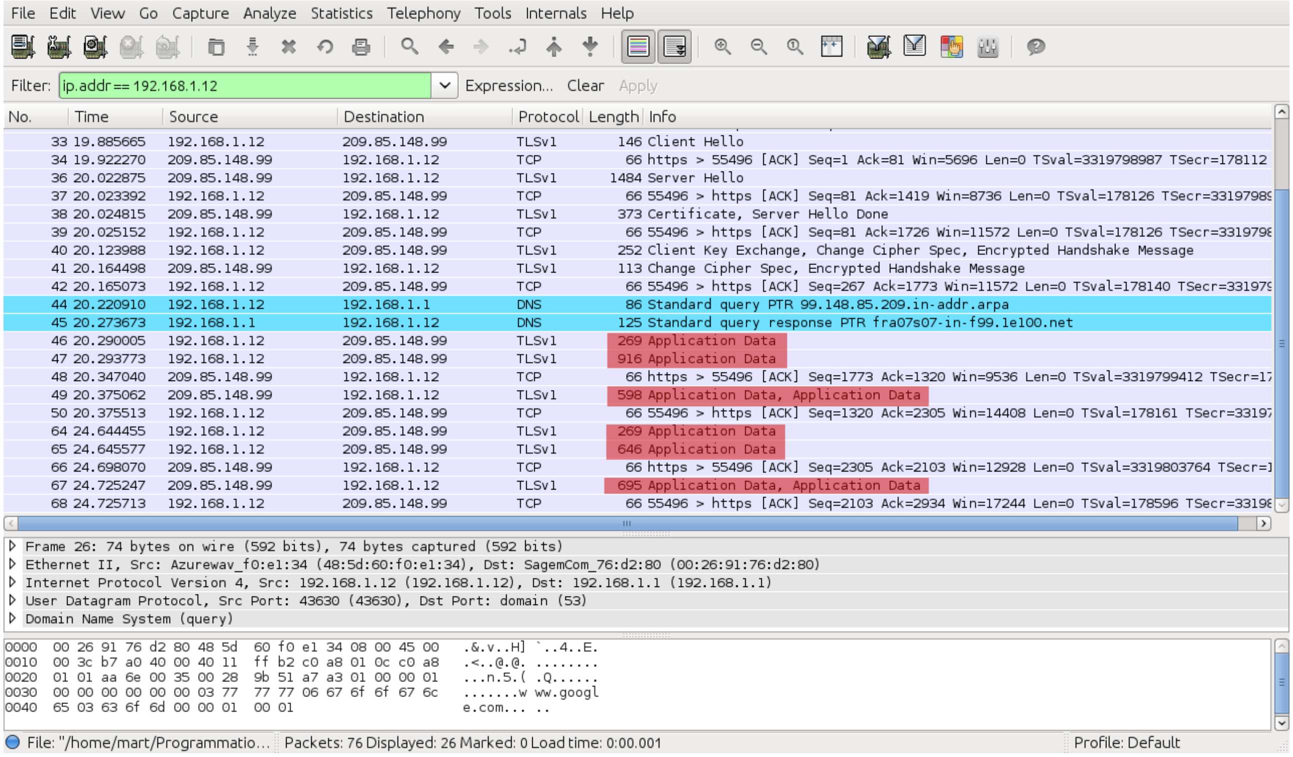
The second experiment used the tool tcpdump to correlate a request with the content of the cache files. The application *LocateMe* has been developed to make a simple location request using the network resources.

1. Start in *blank state*[7]
2. Start tcpdump
3. Activate the wireless
4. Request a location with the application *LocateMe*
5. Stop tcpdump once a location received

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 pattern was derived :

- Two requests are made, one for the cells, one for the wifi.
- The first packets contains every time 269 bytes.
- The uploaded data size is larger than the downloaded.

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


File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Expression... Clear Apply

Filter: ip.addr == 192.168.1.12

```
No.  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=0 TSval=3319798987 TSecr=178112
  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 Len=0 TSval=178126 TSecr=33197989
  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 Len=0 TSval=178126 TSecr=3319798
  40 20.123988  192.168.1.12   209.85.148.99  TLSv1     252 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
  41 20.164498  209.85.148.99  192.168.1.12   TLSv1     113 Change Cipher Spec, Encrypted Handshake Message
  42 20.165073  192.168.1.12   209.85.148.99  TCP        66 55496 > https [ACK] Seq=267 Ack=1773 Win=11572 Len=0 TSval=178140 TSecr=331979
  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-f99.1e100.net
  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=9536 Len=0 TSval=3319799412 TSecr=17
  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=14408 Len=0 TSval=178161 TSecr=33197
  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=12928 Len=0 TSval=3319803764 TSecr=1
  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=17244 Len=0 TSval=178596 TSecr=33198
```

```
▷ Frame 26: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
▷ Ethernet II, Src: Azurewav_f0:e1:34 (48:5d:60:f0:e1:34), Dst: SagemCom_76:d2:80 (00:26:91:76:d2:80)
▷ Internet Protocol Version 4, Src: 192.168.1.12 (192.168.1.12), Dst: 192.168.1.1 (192.168.1.1)
▷ User Datagram Protocol, Src Port: 43630 (43630), Dst Port: domain (53)
▷ Domain Name System (query)
```

```
0000  00 26 91 76 d2 80 48 5d  60 f0 e1 34 08 00 45 00   .&.v..H] `..4..E.
0010  00 3c b7 a0 40 00 40 11  ff b2 c0 a8 01 0c c0 a8   .<..@.@. ........
0020  01 01 aa 6e 00 35 00 28  9b 51 a7 a3 01 00 00 01   ...n.5.( .Q......
0030  00 00 00 00 00 00 03 77  77 77 06 67 6f 6f 67 6c   .......w ww.googl
0040  65 03 63 6f 6d 00 00 01  00 01                     e.com... ..
```

File: "/home/mart/Programmatio... Packets: 76 Displayed: 26 Marked: 0 Load time: 0:00.001 Profile: Default

Figure 1.3: Example of tcpdump capture while a location request

- The size of the downloaded data is directly linked to the number of entries in the cache files.

The figure 1.3 shows an example of collected trace confirming the derived pattern. These observations do not allow to validate the suppositions concerning the packet contents but the patterns are coherent to the model explained before.

1.3 Privacy concerns

1.3.1 Google Cars

In May 2010, Google admitted to German authorities having collected more than what it was supposed to. In addition to access point 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...[8]. In reaction, the data collected was asked to be deleted and the CNIL (independent French

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

administrative authority) fined Google with €100.000[9].

1.3.2 _nomap

Some users considered the automatically 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 scepticism and did not satisfied 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 wanted 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 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.

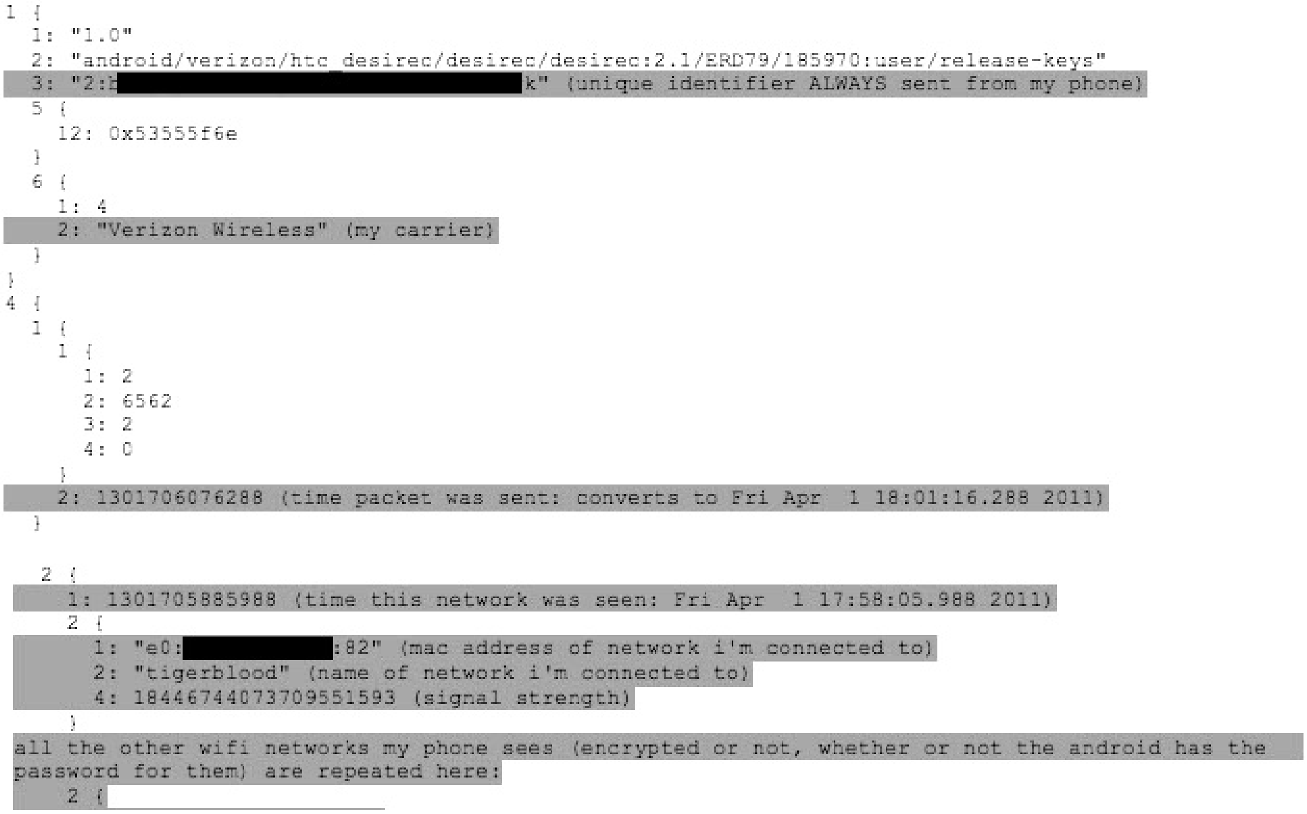
1.3.3 Research of Samy Kamkar

To reply to privacy concerns, Google ensured “The location information sent to Google servers when users opt in to location services on Android is anonymized and stored in the aggregate and is not tied or traceable to a specific user”[7]. The security researcher Samy Kamkar has also looked at the location requested.

He succeeded to decrypt the request made to Google servers and realized that it contains a unique identifier[5]. The figure 1.4 is the content of a request decrypted by Samy Kamkar sent to CNet. 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 anonymized due to the presence of this identifier.

Personal researches showed that this string is contained inside a file next to the cache database and it is renewed every time the *Use wireless networks* option is toggled. 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 regulary.

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



```
1 {
  1: "1.0"
  2: "android/verizon/htc_desirec/desirec/desirec:2.1/ERD79/185970:user/release-keys"
  3: "2:[redacted]k" (unique identifier ALWAYS sent from my phone)
  5 {
    12: 0x53555f6e
  }
  6 {
    1: 4
    2: "Verizon Wireless" (my carrier)
  }
}
4 {
  1 {
    1 {
      1: 2
      2: 6562
      3: 2
      4: 0
    }
    2: 1301706076288 (time packet was sent: converts to Fri Apr  1 18:01:16.288 2011)
  }

   2 {
     1: 1301705885988 (time this network was seen: Fri Apr  1 17:58:05.988 2011)
     2 {
       1: "e0:[redacted]:82" (mac address of network i'm connected to)
       2: "tigerblood" (name of network i'm connected to)
       4: 18446744073709551593 (signal strength)
     }
all the other wifi networks my phone sees (encrypted or not, whether or not the android has the
password for them) are repeated here:
     2 {
```

Figure 1.4: Decrypted request content reveal to CNet by Samy Kamkar

Chapter 2

Security under Android

Introduction

As the number of smartphones is in constant raise, concerns about the security of the system appears. Paradoxically, the users tends to store more and more personal informations on their smartphone 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 sold. Android runs on top of a Linux kernel which is reputed to be virus free.

The aim of this chapter is to explain in detail the actual security mechanisms used to protect the users against malicious applications. Knowing that, a user should be able to reduce his infection risk by adopting simple security principles. Is also explained the different procedures to install an application on a device with the risks associated. 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 inside 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 enable. These tasks are, for example, accessing the current location of the user, update the address book, use 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 Android methods and resources. Without a permission, an application can not access to certain resources or method in the Android system.

2.1.1 Technical details

In appendix 3.10, the list of available permissions are mentioned. 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 the 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 informations (as these towers have a unique identifier, this lower level information could also be used to locate the user[1]).

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 purpose. For example, the permission `READ_PHONE_STATE` has many purpose. It allows an application to be aware 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 provide the access to methods allowing to retrieve the caller phone number. This

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

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 development. These kind of applications require 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 microSD card of the device) for advertising and score storing. Using these permission, it could upload the full content of the SD card (which may contains personal informations from the other applications) to a server. Only a deep analysis such as network monitoring can detect malicious behaviour of an application.

Finally, if a user disagree with the need of a suspicious permission, it has no other choice than not install 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 Applications installation

Unlike iOS where the App Store is the only permitted source of applications[2], the Android operating system propose several way to install an application.

2.2.1 Play Store

By default[3], the Android Play Store (previously named Android Market before its merging with Google Music) is the only source of application. Once a Google account associated to the user's phone, it can use the application Google Play Store which list the available applications and install it quickly. The figure 2.1 shows an example of the interface of the application.

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

- warning when an update is available
- control by Google to avoid malwares
- user comments and review
- paid system with Google Checkout

[2] Alternative markets and applications distributions exists on iOS but they require jailbreaking which is not allow by Apple

[3] 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 not and compatible to approved devices only.

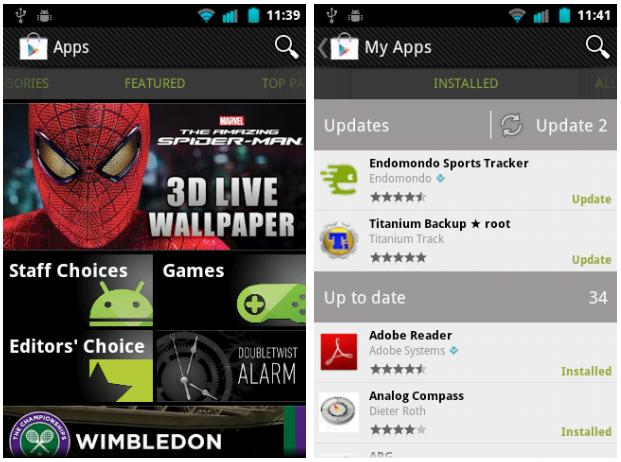


Figure 2.1: Google Play Store interface

Even when distributed by the Play Store, the user should check the asked permission carefully. This is valid at each update where the permissions can change. Several cases of malicious applications on the Play Store have been detected in the past (see bellow). If the approval by Google of each application is a considerable protection it is not perfect and the Play Store should not be considered as a fully safe application distribution medium.

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 application without any user interaction requiered. A simple notice is displayed on the phone once the application is installed.

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[4]. There is no control before the publication of an application.

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. The developed application *DroidWatcher* is an example of what would be possible with a focus on the geolocalization.

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

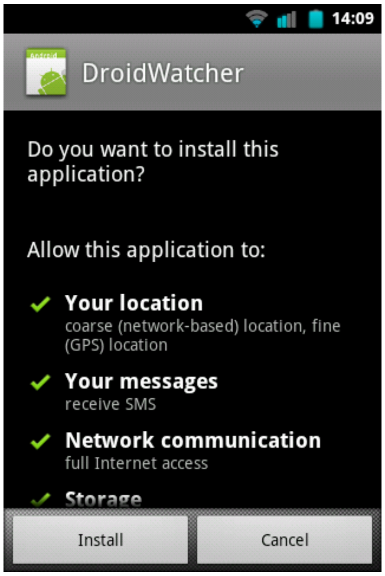
2.2.2 Other sources

By default, installing applications from other source than the Play Store is disallowed. 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. The 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.



DroidWatcher

Do you want to install this application?

Allow this application to:

✓ **Your location**
coarse (network-based) location, fine (GPS) location

✓ **Your messages**
receive SMS

✓ **Network communication**
full Internet access

✓ **Storage**

Figure 2.2: Permissions requiered to install DroidWatcher

An apk file is produced after the compilation of a program and is often proposed on small projects or for beta versions. Also, websites have appear as proposing apk files of non-free applications on the Play Store. These applications should be considered as are the warrez websites on the Windows operating system: highly risky in term of malwares propagation, these applications could have been manipulated to inject malicious code.

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 centralized 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...).

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*[5] developed by Amazon.com Inc. has an approval process to publish applications[6].

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[7], an application can be installed in a few seconds from a computer without any notification on the device connected to a computer (connection done typically using a usb cable). The device does not need to be unlocked to allow the installation.

2.3 Malwares

2.3.1 The DroidDream malware

In spring 2011, a malware named *DroidDream* has widely spread across the Android devices. The particular of this malware was that he used the official Android Play Store (called Android Market at that time). The attackers created several developers accounts and malicious applications (above 50 different applications were detected) on the Play Store. The applications took the name of popular applications or 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[8]) 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 act as a botnet. Due to its ability to install other applications, the Kaspersky Lab's analysts suppose it could have been monetized in the future to be used as a botnet for example.

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

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

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

[8] Data collected from `https://developer.android.com/about/dashboards/index.html`

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 devices[9]. Variants of this malware called DroidDream Lite have been detected a few months later.

2.3.2 General malware type

There is two main types of Android malwares:

The one that uses security flows as DroidDream did. This kind of malware is possible due to the slow update process. The manufacturers tends to provide only a limited number of version updates if any[10]. 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 big range of devices. When a security flow is discovered and a patch published, only a very small percentage of users beneficiate from the patch through an update in the months following the flow discovered. Malware writers can them wrote programs taking advantage of that flow.

The second kind of malware take advantage of the lack of suspicions from the users and simply ask for permissions permitting the malicious behaviour. This is usually the case for applications sending text messages to overtaxed number or stealing 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].

2.3.3 Protection

Observing the large increase of malware applications on the Android platform[13], 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 software would be efficient on antivirus using flows and

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

[10] 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`

[11] Discovered by decompiling the downloaded application from the Android Play Store

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

[13] 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`

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. The open politic on the Google Play Store help the propagation of malwares as the users wrongly suppose a higher control (which is done instead by the users a posteriori).

The Pdroid application[14] 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). Also, for each application, a notification can be launched at the time the resource granted by a permission is used. This feature can 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 requirement 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.

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

Chapter 3

DroidWatcher

3.1 The aim behind DroidWatcher

In the chapter 1, it has been explained how the localization works in the Android devices. What are the methods and how work the localization using wireless. In the chapter 2, it has been explained how an application could behave through the Android system. What is the control of the user, what are the limits of the permission system and how malwares abuse from the user confidence.

DroidWatcher is an application that request several permissions including the one to access the user location. Using this permission the application can record the user permission at all time and monitor its movements. The application can be also controled via text messaging invisibly and sends the recorded location over the internet to a remote server. The application purpose is not to create a malware tracing devices but making the users realising what a device is capable of and how important it is to estimate the risk of malicious behaviour before installing an application. Also having having this application installed could be useful in case of loss or theft of the device.

3.2 Location features

3.2.1 GPS activation

The GPS of a device can be remotely activated (see SMS commands at section). This feature is possible due to a bug discovered in the power control widget[1]. Even if the security flaw has been revealed in April 2010 and a patch released in April 2011, the flaw has been observed as still exploitable on most Android devices running Android 2.3. *TODO: tester sur un Android 4.0*

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

3.2.2 Cell triangulation

The implemented triangulation mechanism using GSM cell towers is inefficient if the device is not connected to the internet. To resolve this constraint, the DroidWatcher application monitors the surroung cell towers identification information and will use this information to compute the location in the futur, the next time the device is connected to internet.

An unofficial API to the Google GSM cell tower database has been discovered and is used to retrieve the location of each cell tower. This database has been selected as it is one of the most complete compared to the other free alternatives.

schema du méchanisme utilisé

This triangulation mechanism is however known as unprecise as explained in the section 3.3.3 about technical difficulties.

3.3 Technical difficulties

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

3.3.1 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 localization process done by DroidWatcher is affected by this idle purpose.

This is the case for example of the GPS that needs to constant update of the position. The GPS will sometime stop monitoring the postion of the user if it can not get a fix[2] on the location of the user. This effect is undependant 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. This solution as however not been 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 logest and more discrete way as possible.

[2] cf section 1.1.1 for the information needed to get a GPS fix

3.3.2 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[3] 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.

And unexpected change introduced in the 4.0 version of Android was the way a device manage the start of an application in background. DroidWatcher has been concieved 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 noticable without analysis. In 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.

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.

3.3.3 Cell tower triangulation

To compute the location of the user, a triangulation algorithm has been developed. This algorithm is however known as unprecise for several reason. To compute the location, the algorithm uses the signal strength captured GSM cell towers nearby. The signal strength is a very fluctuating variable. At a same distance to an cell tower, the signal will variate 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 requiered long monitoring, the compuation and ponderation of the variables has been done based on personal observation. This is known as unprecise but achieve the purpose to be able to record an approximate location at all time when GSM connectivity is available. x

[3] 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.

3.4 User manual

3.4.1 SMS commands

The messages are intercepted before arriving to the message application. If the message contains a defined code, the phone will do a defined 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 won't be affected.

`BIGB` : 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:

- BIGBGPSON
- bigbSetServerhttp://watcher.dotzer.me/collect
- Ping

Examples of incorrect messages

- BIGB GPSON
- Ping!

To easily test if the app is running send the message `PING` it will reply 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.

3.5 What collects 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 requiere any permission and that any application can collect it.

3.6 What collects 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 requiere any permission and that any application can collect it.

3.7 When run the application ?

The application start at the phone boots and when the user unlock its phone. Killing the process will only stop it until the next time the phone is unlocked. Uninstalling the application `DroidWatcher` will fully remove it.

3.8 What is stored on my 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.

3.9 Who can see my location ?

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

3.10 Install application on its own server

To see the information collected, you can install the DroidWatcher collecting website on your own web server. The server use the python framework Django 1.3[4]. The following steps explain the deployement of the application on a Debian Lenny server running Apache and mod-wsgi. The full configuration and securitization 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.

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
   ```

[4] Available at https://www.djangoproject.com/

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

Appendix

List of permissions

String	ACCESS_CHECKIN_PROPERTIES	Allows read/write access to the ”properties” table in the checkin database, to change values that get uploaded.
String	ACCESS_COARSE_LOCATION	Allows an application to access coarse (e.g., Cell-ID, WiFi) location
String	ACCESS_FINE_LOCATION	Allows an application to access fine (e.g., GPS) location
String	ACCESS_LOCATION_EXTRA_COMMANDS	Allows an application to access extra location provider commands
String	ACCESS_MOCK_LOCATION	Allows an application to create mock location providers for testing
String	ACCESS_NETWORK_STATE	Allows applications to access information about networks
String	ACCESS_SURFACE_FLINGER	Allows an application to use SurfaceFlinger’s low level features
String	ACCESS_WIFI_STATE	Allows applications to access information about Wi-Fi networks
String	ACCOUNT_MANAGER	Allows applications to call into AccountAuthenticators.
String	AUTHENTICATE_ACCOUNTS	Allows an application to act as an AccountAuthenticator for the AccountManager
String	BATTERY_STATS	Allows an application to collect battery statistics
String	BIND_APPWIDGET	Allows an application to tell the AppWidget service which application can access AppWidget’s data.
String	BIND_DEVICE_ADMIN	Must be required by device administration receiver, to ensure that only the system can interact with it.
String	BIND_INPUT_METHOD	Must be required by an InputMethodService, to ensure that only the system can bind to it.
String	BIND_REMOTEVIEWS	Must be required by a RemoteViewsService, to ensure that only the system can bind to it.
String	BIND_WALLPAPER	Must be required by a WallpaperService, to ensure that only the system can bind to it.
String	BLUETOOTH	Allows applications to connect to paired bluetooth devices
String	BLUETOOTH_ADMIN	Allows applications to discover and pair bluetooth devices
String	BRICK	Required to be able to disable the device (very dangerous!).
String	BROADCAST_PACKAGE_REMOVED	Allows an application to broadcast a notification that an application package has been removed.
String	BROADCAST_SMS	Allows an application to broadcast an SMS receipt notification
String	BROADCAST_STICKY	Allows an application to broadcast sticky intents.

String	BROADCAST_WAP_PUSH	Allows an application to broadcast a WAP PUSH receipt notification
String	CALL_PHONE	Allows an application to initiate a phone call without going through the Dialer user interface for the user to confirm the call being placed.
String	CALL_PRIVILEGED	Allows an application to call any phone number, including emergency numbers, without going through the Dialer user interface for the user to confirm the call being placed.
String	CAMERA	Required to be able to access the camera device.
String	CHANGE_COMPONENT_ENABLED_STATE	Allows an application to change whether an application component (other than its own) is enabled or not.
String	CHANGE_CONFIGURATION	Allows an application to modify the current configuration, such as locale.
String	CHANGE_NETWORK_STATE	Allows applications to change network connectivity state
String	CHANGE_WIFI_MULTICAST_STATE	Allows applications to enter Wi-Fi Multicast mode
String	CHANGE_WIFI_STATE	Allows applications to change Wi-Fi connectivity state
String	CLEAR_APP_CACHE	Allows an application to clear the caches of all installed applications on the device.
String	CLEAR_APP_USER_DATA	Allows an application to clear user data
String	CONTROL_LOCATION_UPDATES	Allows enabling/disabling location update notifications from the radio.
String	DELETE_CACHE_FILES	Allows an application to delete cache files.
String	DELETE_PACKAGES	Allows an application to delete packages.
String	DEVICE_POWER	Allows low-level access to power management
String	DIAGNOSTIC	Allows applications to RW to diagnostic resources.
String	DISABLE_KEYGUARD	Allows applications to disable the keyguard
String	DUMP	Allows an application to retrieve state dump information from system services.
String	EXPAND_STATUS_BAR	Allows an application to expand or collapse the status bar.
String	FACTORY_TEST	Run as a manufacturer test application, running as the root user.
String	FLASHLIGHT	Allows access to the flashlight
String	FORCE_BACK	Allows an application to force a BACK operation on whatever is the top activity.
String	GET_ACCOUNTS	Allows access to the list of accounts in the Accounts Service
String	GET_PACKAGE_SIZE	Allows an application to find out the space used by any package.

String	GET_TASKS	Allows an application to get information about the currently or recently running tasks: a thumbnail representation of the tasks, what activities are running in it, etc.
String	GLOBAL_SEARCH	This permission can be used on content providers to allow the global search system to access their data.
String	HARDWARE_TEST	Allows access to hardware peripherals.
String	INJECT_EVENTS	Allows an application to inject user events (keys, touch, trackball) into the event stream and deliver them to ANY window.
String	INSTALL_LOCATION_PROVIDER	Allows an application to install a location provider into the Location Manager
String	INSTALL_PACKAGES	Allows an application to install packages.
String	INTERNAL_SYSTEM_WINDOW	Allows an application to open windows that are for use by parts of the system user interface.
String	INTERNET	Allows applications to open network sockets.
String	KILL_BACKGROUND_PROCESSES	Allows an application to call killBackgroundProcesses(String).
String	MANAGE_ACCOUNTS	Allows an application to manage the list of accounts in the AccountManager
String	MANAGE_APP_TOKENS	Allows an application to manage (create, destroy, Z-order) application tokens in the window manager.
String	MASTER_CLEAR	
String	MODIFY_AUDIO_SETTINGS	Allows an application to modify global audio settings
String	MODIFY_PHONE_STATE	Allows modification of the telephony state - power on, mmi, etc.
String	MOUNT_FORMAT_FILESYSTEMS	Allows formatting file systems for removable storage.
String	MOUNT_UNMOUNT_FILESYSTEMS	Allows mounting and unmounting file systems for removable storage.
String	NFC	Allows applications to perform I/O operations over NFC
String	PERSISTENT_ACTIVITY	This constant is deprecated. This functionality will be removed in the future; please do not use. Allow an application to make its activities persistent.
String	PROCESS_OUTGOING_CALLS	Allows an application to monitor, modify, or abort outgoing calls.
String	READ_CALENDAR	Allows an application to read the user's calendar data.
String	READ_CONTACTS	Allows an application to read the user's contacts data.
String	READ_FRAME_BUFFER	Allows an application to take screen shots and more generally get access to the frame buffer data
String	READ_HISTORY_BOOKMARKS	Allows an application to read (but not write) the user's browsing history and bookmarks.

String	READ_INPUT_STATE	Allows an application to retrieve the current state of keys and switches.
String	READ_LOGS	Allows an application to read the low-level system log files.
String	READ_PHONE_STATE	Allows read only access to phone state.
String	READ_SMS	Allows an application to read SMS messages.
String	READ_SYNC_SETTINGS	Allows applications to read the sync settings
String	READ_SYNC_STATS	Allows applications to read the sync stats
String	REBOOT	Required to be able to reboot the device.
String	RECEIVE_BOOT_COMPLETED	Allows an application to receive the ACTION_BOOT_COMPLETED that is broadcast after the system finishes booting.
String	RECEIVE_MMS	Allows an application to monitor incoming MMS messages, to record or perform processing on them.
String	RECEIVE_SMS	Allows an application to monitor incoming SMS messages, to record or perform processing on them.
String	RECEIVE_WAP_PUSH	Allows an application to monitor incoming WAP push messages.
String	RECORD_AUDIO	Allows an application to record audio
String	REORDER_TASKS	Allows an application to change the Z-order of tasks
String	RESTART_PACKAGES	This constant is deprecated. The restartPackage(String) API is no longer supported.
String	SEND_SMS	Allows an application to send SMS messages.
String	SET_ACTIVITY_WATCHER	Allows an application to watch and control how activities are started globally in the system.
String	SET_ALARM	Allows an application to broadcast an Intent to set an alarm for the user.
String	SET_ALWAYS_FINISH	Allows an application to control whether activities are immediately finished when put in the background.
String	SET_ANIMATION_SCALE	Modify the global animation scaling factor.
String	SET_DEBUG_APP	Configure an application for debugging.
String	SET_ORIENTATION	Allows low-level access to setting the orientation (actually rotation) of the screen.
String	SET_POINTER_SPEED	Allows low-level access to setting the pointer speed.
String	SET_PREFERRED_APPLICATIONS	This constant is deprecated. No longer useful, see addPackageToPreferred(String) for details.
String	SET_PROCESS_LIMIT	Allows an application to set the maximum number of (not needed) application processes that can be running.
String	SET_TIME	Allows applications to set the system time
String	SET_TIME_ZONE	Allows applications to set the system time zone
String	SET_WALLPAPER	Allows applications to set the wallpaper
String	SET_WALLPAPER_HINTS	Allows applications to set the wallpaper hints

String	SIGNAL_PERSISTENT_PROCESSES	Allow an application to request that a signal be sent to all persistent processes
String	STATUS_BAR	Allows an application to open, close, or disable the status bar and its icons.
String	SUBSCRIBED_FEEDS_READ	Allows an application to allow access the subscribed feeds ContentProvider.
String	SUBSCRIBED_FEEDS_WRITE	
String	SYSTEM_ALERT_WINDOW	Allows an application to open windows using the type TYPE_SYSTEM_ALERT, shown on top of all other applications.
String	UPDATE_DEVICE_STATS	Allows an application to update device statistics.
String	USE_CREDENTIALS	Allows an application to request authtokens from the AccountManager
String	USE_SIP	Allows an application to use SIP service
String	VIBRATE	Allows access to the vibrator
String	WAKE_LOCK	Allows using PowerManager WakeLocks to keep processor from sleeping or screen from dimming
String	WRITE_APN_SETTINGS	Allows applications to write the apn settings
String	WRITE_CALENDAR	Allows an application to write (but not read) the user's calendar data.
String	WRITE_CONTACTS	Allows an application to write (but not read) the user's contacts data.
String	WRITE_EXTERNAL_STORAGE	Allows an application to write to external storage
String	WRITE_GSERVICES	Allows an application to modify the Google service map.
String	WRITE_HISTORY_BOOKMARKS	Allows an application to write (but not read) the user's browsing history and bookmarks.
String	WRITE_SECURE_SETTINGS	Allows an application to read or write the secure system settings.
String	WRITE_SETTINGS	Allows an application to read or write the system settings.
String	WRITE_SMS	Allows an application to write SMS messages.
String	WRITE_SYNC_SETTINGS	Allows applications to write the sync settings

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

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

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

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

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