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

I gratefully acknowledge my supervisor Professor Gildas Avoine for his constant support, explanations, corrections and trust during this year of work.

I also thank the UCL for providing me the required material for the security analysis of the camera and Xavier Carpent and Vianney Le Clément who accepted to be readers of this thesis.

Finally, I would like to thank Bernard for his help and advises for the redaction of this document and as well as Antoine, Céline, Charlotte and Ieva for their time spent reading and correcting this thesis.

Contents

Introduction

Background

Extract from article "The Really Smart Phone" in the Wall Street journal (April 22, 2011) [10].

> [...] As a tool for field research, the cellphone is unique. Unlike a conventional land-line telephone, a mobile phone usually is used by only one person, and it stays with that person everywhere, throughout the day. [...]
>
> Advances in statistics, psychology and the science of social networks are giving researchers the tools to find patterns of human dynamics too subtle to detect by other means. At Northeastern University in Boston, network physicists discovered just how predictable people could be by studying the travel routines of 100,000 European mobile-phone users. [...]
>
> After analyzing more than 16 million records of call date, time and position, the researchers determined that, taken together, people's movements appeared to follow a mathematical pattern. The scientists said that, with enough information about past movements, they could forecast someone's future whereabouts with 93.6% accuracy. The pattern held true whether people stayed close to home or traveled widely, and wasn't affected by the phone user's age or gender. [...]

We introduce the background of the current thesis with this extract of an article published in 2011 in the Wall Street Journal. This gives a rather frightening view on the potential exploitation of the massive amount of data made available from smartphones. One year after the publication of this article, one can only confirm this structural trend.

The usage of technology has become an integral part of human life. It keeps us organised and it enables our ubiquitous interactions with others in a way that is becoming less and less conscious. Even more we get so used to the convenience of these technologies that we might get not very reluctant to unveil a part of our private life. As users seem to be unable to protect their privacy, governements make recommendations[1] and privates companies develop certifications[2]. This is clearly a new way of life.

[1] In March 2012, the US Federal Trade Commission has published recommendations for businesses and policymakers to protect consumers' privacy `http://www.ftc.gov/os/2012/03/120326privacyreport.pdf`

[2] For instance, TRUSTe is a private company certifying websites and applications comply with a set privacy requirements.

This thesis does not focus on computers or websites on the Internet but on the devices implementing technologies that contributes to our new way of living. These devices are designed and marketed more as easy-to-use appliances than as computers with their complexity and known vulnerabilities. Actually a modern smartphone is not an appliance but a general purpose computer fitting in a very small and convenient case. From the end-user perspective the privacy aspect of such a device is usually ignored or considered as good enough. Also the end-user confidence in the manufacturer's concern to deal with privacy is a recurrent problem as indicated by various privacy scandals[3].

Case study choices

Although there are many very important and interesting aspects (marketing, social and legal aspects to name some) related to this new "world", this thesis will look only at technical capabilities and related weaknesses of selected case studies. As part of this thesis, it has been decided to focus on two types of devices: the Android smartphone operating system and wireless cameras.

Smartphone is an obvious candidate for reasons mentioned above. The smartphone penetration rate is in constant rise and, in a few years, it is expected that the majority of people will own such a device in developed countries. Similarly to our personal computers, these devices contain more and more personal data. The consequences of compromising such data can be very severe for an end-user. The cause of data breach can be both related to potential security weaknesses or to abuses for marketing or criminal purposes (eg: identity theft).

Wireless camera is an interesting candidate because much less mentioned in literature while privacy concerns related to it are real. The consequences of camera compromising and unauthorised access to the video stream is potentially very dangerous and can have the opposite effect than the one expected (to ease abuses instead of protecting). In order of demonstrating vulnerabilities and privacy consequences, a specific case has been selected: the D-Link DSC 2130 camera.

Potential abuses

Beyond the two study cases, several technologies face potential abuses related to the privacy of the users. This section does not intend to cover every technology but to give a brief presentation for reflection on the risks once these technologies are adopted for our everyday use.

Mobile phones

As the Wall Street Journal extract introduced, the amount of data collected using mobile phones reveals patterns in the behaviour of human beings. While this

[3] In April 2011, a cache file containing 8 months of previous user locations has been discovered on iPhones, Section 2.2.3 explains the purpose of such file and its presence on Android.

research focused on voluntary tracked family using smartphones, there exists many other ways of tracking a mobile phone.

For instance, a phone operator is capable of locating any user in a range of a few hundreds of meters. This localisation is possible using the information from the GSM cell towers on which a user is connected. This is a requirement imposed by the US and European directives E911 and E112 in order to be able to locate a mobile phone owner in a case of emergency [18].

In March 2011, the German politician Malte Spitz took legal proceedings against Deutsche Telekom mobile phone company to hand over the data collected from his phone. By combining the received data (35,831 spreadsheet rows), he was able to create a map representing his movements during the six months of collected data, including the time of each phone call and text message sent or received [19].

RFID

RFID tags are present in more and more products, from clothes to credit cards or passports, their uses ranges from inventory tracking to identification or payment transactions. As long as the distance from the reader is sufficient (from a few centimetres to meters depending on the tag), an RFID tag enables the transmission of information without any interaction needed from the person who carries the tag. The data transaction can even happen without the user realising it. Due to this last point, RFID technology is often involved in privacy issue scandals.

The criticism of RFID is based on the possible traceability issues as well as the privacy concerns. Privacy advocates ask for the destruction of RFID tags when not in use (for instance after buying an item where the tag has been used for inventory purpose) or advise covering it with a aluminium foil.

RFID tags are now used in credit cards for contactless payments. In January 2012, Kristin Paget demonstrated[4] that, with compatible RFID reader, it is possible to retrieve sensible bank information from the card such as credit card number and expiration date, along with the one-time CVV number used by contactless cards to authenticate payments.

Wifi hotspot

More and more companies or internet providers offer wifi sharing features. For instance, it is the case with the international FON[5] system and, at a local level, the wireless system deployed in the centre of Louvain-La-Neuve for the students and members of the UCL[6]. These technologies are very convenient as they allow to access the Internet from outside the home network.

However these technologies incorporate a security risk. For instance, it is possible to create a fake access point on which victims may connect and reveal, in the best

[4] Presentation of the hack was done during the Shmoocon 2012 conference, slides available at `http://www.tombom.co.uk/Paget-shmoocon-credit-cards.pdf`

[5] FON network `http://corp.fon.com/`

[6] Université Catholique de Louvain, Belgium `https://www.uclouvain.be/en-wifi.html`

case, their credentials for the access point or more sensitive information during the navigation on the Internet. During the master in computer sciences at the UCL, the pattern of such an attack has been demonstrated at the occasion of a security. A fake FON access point was created to retrieve FON credentials as well as passwords during the surf using inexpensive material.

Document organisation

The thesis is divided into two distinct parts.

Part I deals with the Android system and is composed of four chapters.

In **Chapter 1**, focuses on Android system as a research subject, also the problem of localisation is discussed. In **Chapter 2**, the localisation techniques used in Android and the related privacy concerns are reviewed. In **Chapter 3**, an overview of the security mechanisms currently used in the Android system to avoid abuses from the applications is presented. Also it is discussed how the current security weaknesses are exploited by malicious developers. **Chapter 4** is a presentation of *DroidWatcher*, the software developed during this thesis as an implementation of the concepts introduced in the previous two chapters.

Part II concerns wireless cameras and is composed of two chapters.

Chapter 5 is an introduction to wireless cameras. It presents the different types of cameras and the risks of discovery and compromising of a digital web camera. In **Chapter 6**, a full security analysis of the chosen camera model D-Link DCS-2130 is presented.

Chapter 7 is the conclusion of the work, summarising the observations made from the analysis and the issues detected.

The appendix provides the documentation related to softwares developed in the scope of this thesis.

Licence

The author of this work believes in the importance of free licences for the development of technology and sharing of knowledge. Accordingly, the content of this thesis is released under various open licences.

The text of the current report is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License. This license allows you to copying, distribution, transmission, adaptation the work and making commercial use of the work under the some conditions. Details and the full licence text are available at `https://creativecommons.org/licenses/by-sa/3.0/`.

The code developed and published in the context of this thesis is, with exception of explicit mention of otherwise, licensed under the GNU GENERAL PUBLIC

Part I

Android

Chapter 1

Introduction to Android

This part is composed of three chapters. Consecutive order of chapters should be observed when reading this part. However, technical details can be skipped without losing the general understanding.

The chapter on the **localisation using Android** presents the different methods available to retrieve information regarding the localisation of smartphones. The aim is to define the capabilities of an application in terms of localisation. These methods are often unknown and considered as a black-box but when studied they reveal to privacy concerns. The anonymity and the traces left by the localisation requests are examined in this chapter.

The **security of Android** chapter focuses on the capabilities of an application in terms of security. The permission model implemented in Android is explained focusing on 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 propagate to a user device. The malware risk on Android has greatly increased in the past months on Android and the security of the platform is a recurring concern. This chapter intends to clarify the risks by explaining the possible abuses that are currently known at this day.

The combination of the first two chapters aims at developing a good understanding of the mechanisms of localisation and related security issues on an Android application. As an example of practical implementation of these concepts, the **DroidWatcher** mobile application has been developed. This application aims at demonstrating 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. This important market combined with the increasing smartphone presence, it is very likely that a mobile device is powered by this operating system now and in a reasonable 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[1].

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 enables more abusive usages and may lead to an increased risk of malicious application propagation. In August 2012, Kaspersky Lab published a report mentioning that the number of Trojans targeting the Android platform nearly tripled from the first quarter of the year [14]. 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, it is believed that the result of this research are relevant to other systems as well. Today, every smartphone system has localisation capabilities similar to the ones used in Android and malicious applications could abuse from these capabilities without the user realising it. The malicious application propagation is directly depending on the platform[2], but the risks exist and potentially harmful applications are present on all systems that enables such abusive behaviour.

Why the localisation?

Nowadays, the localisation privacy is often neglected. Applications such as Foursquare (which are intend to actively and consciously publish one's current position on social networks) are very popular nowadays[3]. In February 2010, J.Y. Tsai, P.G. Kelley, L.F. Cranor and N. Sadeh published an report about 89 applications (include 63 present on mobiles) that used a location-sharing feature similar to Foursquare [13]. The large number and popularity of these applications

[1] Market share statistics according to Mobile Statistics `http://www.mobilestatistics.com/mobile-statistics/`

[2] The difference of an application publication on the official distribution medium between Android and iOS is studied in Section 3.3.1

[3] Foursquare had over 20 million of users in April 2012 according to `https://foursquare.com/about/`

is a clear sign of this tendency.

This lack of focus on privacy has been understood and exploited by advertisers and developers. Mobile technology brings a new dimension to the constant search for knowing more about people: the *Where am I?*. The user profiling is critical for an efficient targeted advertisement (eg: Amazon suggesting books that might interest the visitor). The localisation possibilities are an important criteria for this profiling and very valuable for marketing purposes.

However, localisation process can be used in a more precise and worrying way than for targeted advertisements (cf the Wall Street Journal article). A location-aware application is capable of a continuous background monitoring of an Android phone. Demonstrating these possibilities in a transparent way is the aim behind the DroidWatcher application.

The location aspect is only studied regarding the capabilities of requesting the location of a device. With the exception of the cache file described in Section 2.2.1, the forensic inspection of an Android device has not been studied and could be included in future researches.

Chapter 2

Localisation using Android

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 enables, for example, to show directly 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 use of an application in a particular region.

The details of the localisation methods used by the Android system are often unknown or unclear. Several techniques exist 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 to have a better understanding of the localisation system that mostly works as a black box.

2.1 GPS

2.1.1 GPS signal

The GPS, or Global Positioning System, is a technology based on satellite trilateration. GPS satellites are navigating around the Earth in a way to maximise the number of visible satellites anywhere at any time. There are currently 31 working satellites in orbit [8]. 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 2.1. 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).

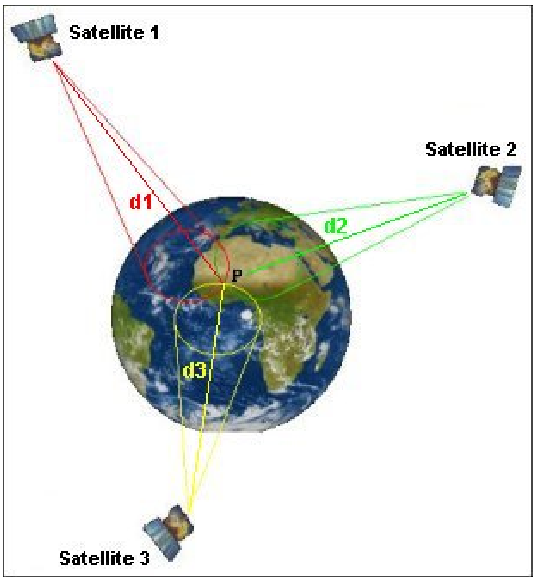


Figure 2.1: Signals from multiple satellites are required to calculate a position
Copyright education.fr

The accuracy of this method depends on the surroundings of the user. In clear sight, the GPS is accurate to a few meters but the accuracy 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 [9]. To improve the TTFF, GPS embedded into smartphones often use assisted-GPS technology (aGPS) by acquiring almanac and ephemeris data over a fast network connection when available.

2.1.2 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) were able to 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 incapable of producing Selective Availability [7].

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

2.2 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 metres, the localisation is faster than using GPS, it works indoors, consumes less battery power than a GPS receiver chip and one access point is enough to locate a device.

2.2.1 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 [12].

Companies have quickly understood the value of this information and the economical interest of having their 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 use 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 2.4.

2.2.2 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 feeding the Google Street View database. In addition to that, the cars are also wardriving[4]. Having a GPS module on the car and mapping almost every main city in the world, it was a good opportunity to constitute a very accurate database.

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

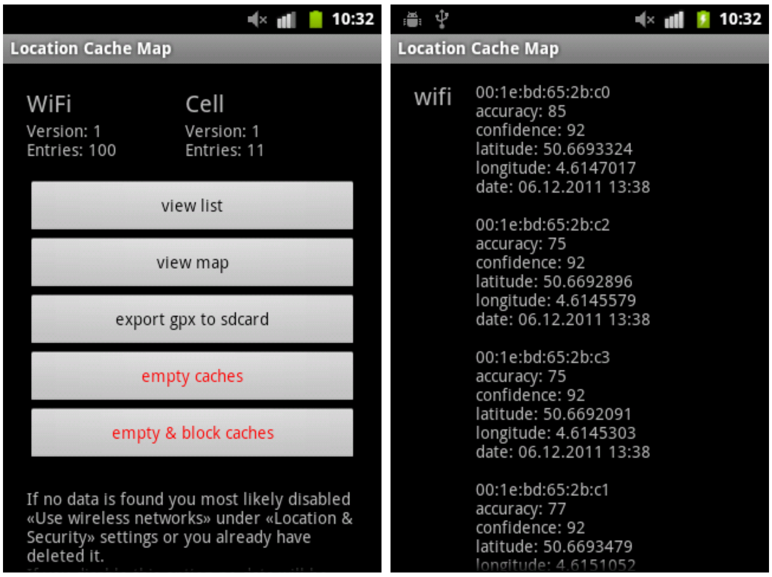
[3] Car equipped with a GPS, wireless and cell tower receiver collecting data in the streets

[4] Wardriving is the activity of driving through the streets of a city to collect information such as wireless access points location.

As most Android devices are equipped with a GPS receiver, collecting via crowdsourcing is also possible. When an Android device has both GPS and wifi capabilities, while using application such as Google Maps, the location of the user and surround wifi and cell tower data are uploaded to Google's servers. This way, the database of wireless access points and cell towers is always up-to-date[5].

2.2.3 Location cache files

Previous cell tower and access point locations are stored in an unencrypted system file. 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 2.2.



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 2.2: Captures from the application Location Cache by Remy Demy

The recent criticism regarding 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[6]), 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 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 enable 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 that has the correct permissions. The DroidWatcher application relies 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 [3] to parse the content of these two files and produce a GPS trace file in GPX

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

[6] iOS 4.3.3 Software Update `https://support.apple.com/kb/dl1358`

format[7] representing the approximate movement of the device. However, a root access to the phone is required to access these files[8], this could prevent 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.

2.3 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 enables 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 2.2 and will not be discussed here. The location system in Android usually uses both wireless access points and cell towers location methods simultaneously and indifferently for the user and developer. The advantage of using cell tower location over the wireless is the fact that it allows to locate a device with an approximate accuracy of 1 km almost everywhere.

2.4 Privacy concerns

2.4.1 Google Cars

In May 2010, Google admitted to German authorities having collected more data than what it was supposed to. In addition to access points 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...[9]. In reaction, the data collected was asked to be deleted and the CNIL (independent French administrative authority) fined Google with €100.000[10].

2.4.2 _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 Car or an Android device, the access point is removed from the database. Google hopes

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

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

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

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

than over time, the _nomap string will be adopted by other location providers [4].

This proposition was received with much 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 his access point while people want a way to explicitly opt in instead. Many people that are concerned with 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 his access points from commercial databases.

2.4.3 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" [2]. The security researcher Samy Kamkar has also looked into the location requested.

He succeeded in decrypting the request made to Google servers and found that it contains a unique identifier [16]. The identifier is unique to the cell phone and present in every request. Even though 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 to trace him. He affirms there is no proof the location is anonymised due to the presence of this identifier.

2.5 Personal researches

Several facts about the storage of information and privacy have been announced. To verify these facts, further investigations have been carried out as part of this research. The applications used to implement the analyses are presented in the appendix B.

To carry out the experiments, a rooted smartphone using Android 2.3 was used. As most of the location information, including the cache databases files, is 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.

About root access

On most devices, the owner of an Android device has only a limited access to the system. He 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 one device to another since root access protection is left to manufacturers' choice and can require a complex procedure[11]. Gaining root access is considered sufficient to void the warranty of a device with

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

many manufacturers.

When enabled at the user level, it is possible for an application to ask as well to gain root access and access most of the device capabilities. However these applications still need the user authorisation to do so, gaining root access does not decrease fundamentally the security regarding applications security.

To carry out the experiments described in this section, a root access was necessary to be able to monitor every aspect of the system.

2.5.1 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 question is 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

In order to carry out the experiment, the two scripts `androdump.py` and `compare_dump.py` have been developed and are available on the appendix 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 (alike battery state or statistic files) 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.

2.5.2 Experiment 2: Impact of location requests

Goal

When a location is requested on an Android device, the system sends an encrypted request to Google's location servers. 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.

2.5.3 Experiment 3: Correlation between size and cache of requests

Goal

As the request used to locate the device using wifi and cell towers 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 attempted to determine the patterns in the requests form to correlate a request with the content of the cache files.

Methodology

1. Start in *blank state*[12]
2. Start monitoring the traffic

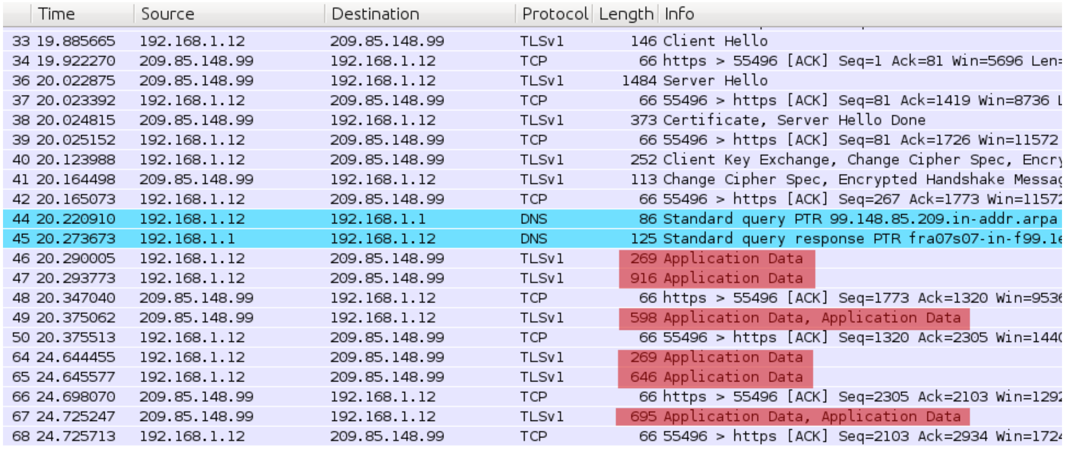
[12] *Blank state* : wireless turned off, empty location cache, location permission turned off, no process requiring the location such as Google Maps running.

3. Activate the wireless

4. Request the current location

5. Stop monitoring the traffic once a location received

The network monitoring has been made with the tool *tcpdump*[13] installed on the Android device.



```
   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 2.3: Example of tcpdump capture while a location request displayed in Wireshark

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 repetition of the experiment (results displayed in Table 2.1), the following observations were made. Each location request is composed of two communications with Google's servers. In each communication, a first packet of 271 bytes is sent followed by another one of variable 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
4 (1)	9	271+463/632	271+898/861	2 (1)	9
6 (1)	10	271+434/619	271+916/873	1	10

Table 2.1: Example of location requests and effect on the cache files

Table 2.1 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 are six surrounding GSM cell towers detected but only one is valid[14]. **S. wifi** represents the number of surrounding wireless access points at the

[13] TCPDUMP 4.3 `http://www.tcpdump.org/`

[14] 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 often have a very weak signal strength.

collect time. **Req 1** represents the first communication and **Req 2** the second one ($271 + 644/654$ 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 brackets represents the number of valid entries in the cache file, as the experiment starts in blank state, the cache files are empty at start).

Figure 2.3 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's 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 enable us to validate the suppositions concerning the content of the packet but the patterns are coherent to the model explained before.

2.5.4 Experiment 4: Unique identifier

Goal

As mentioned in Section 2.4.3, Samy Kamkar observed that a unique identifier was used in the requests made to Google's 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

The content of the locations cache folder in the system file was observed and this identifier was found inside the file `gls.platform.key` next to the cache databases. 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 created. The content of the identifier file `gls.platform.key` is different to the previous value every time the option is toggled.

Conclusions

The traceability is limited due to this constraint. Although it is relatively easy to change this value, it is quite clear that very few users are aware of the existence of this value and will apply this manipulation regularly.

Chapter 3

Security of Android

As the number of smartphones is constantly rising, the level of concerns about the security of the system increases. Paradoxically, users tend to store more and more personal data on their smartphones and are not aware of the security issues of such devices. Malware have been discovered in 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 associated risks are evaluated. The different procedures from the publication to the installation of an application on a device are particularly examined.

The forensic aspect of information retrieval from a device without the owner's consent has not been analysed here.

These clarifications are essential to understand the limits and possibilities for the developed *DroidWatcher* application (see Chapter 4) to be effective.

3.1 Permissions

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

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

3.1.1 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 3.1: 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 3.1 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 user receives a notification of the crash of the application.

The permission processed is conceived to control the access to information and to not a phone characteristic. For instance, there is no permission to use the GPS but one to access a precise location whatever the source of the information. The `ACCESS_COARSE_LOCATION` permission is not limited to the use of the high level `LocationManager` methods but it 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]).

3.1.2 Weaknesses

The way the permission system is implemented does not fully prevent malicious behaviour. The permission description is unclear and can have different purposes. For example, the permission `READ_PHONE_STATE` is required for many actions. It enables an application to be notified when a phone call is processed or when the device is locked, it also gives information about the unique identifier of the phone 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 a type of information leakage that could have been avoided.

[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

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. This kind of applications requires the permission to access the Internet in order 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. By using these permissions, it could upload the full content of the SD card (which may contain personal information from the other applications) to a server. Only an in-depth analysis such as network monitoring can detect malicious behaviours of an application.

Finally, if a user disagrees with the need of a suspicious permission, he has no other choice than not to install the application. There is no possibility to partially accept the permissions. Considering this restriction, it is clear than in most cases, the user will accept any permissions without regards of their type, in order to use the particular application.

3.2 Installation of applications

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

3.2.1 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 linked to the user's phone, the user can use the Google Play Store application which lists the available applications and allows quick installation. Figure 3.1 shows an example of the interface of the application.

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
- Users' comments and reviews
- Payment 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 major 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

[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.

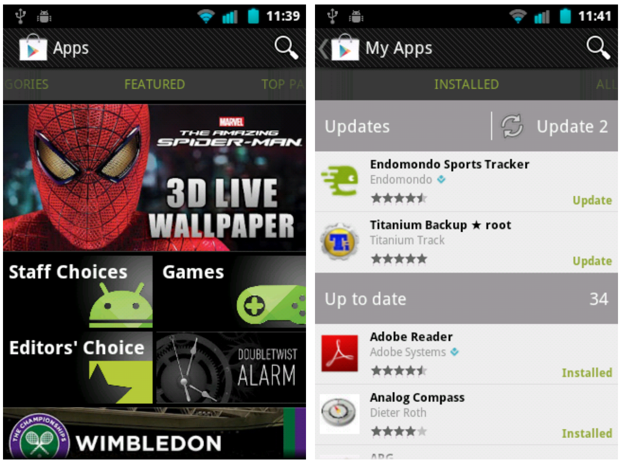


Figure 3.1: Google Play Store interface

is installed.

3.2.2 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

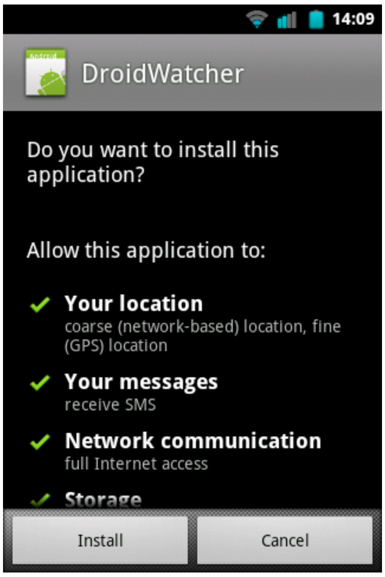
The *.apk* extension is the convention for installable applications on the Android operating system in the same way as *.deb* or *.rpm* are on Debian and Fedora operating systems.

A user trying to open such files on a device launches the installation process in the same way as if the Android Play Store would be used. The required permissions are displayed and require the confirmation of the user. Figure 3.2 shows the permission screen when a user tries to install the application DroidWatcher. The same screen will appear while either using 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 marketplaces

In the same way as the Android Play Store, several alternative market places exist. Alternative market places enables downloading apk files from a centralised interface. It is considered to be an alternative to the Google Play Store with the advantages of a centralised distribution medium (payment system, users' reviews, moderation, etc.). Manufacturers or service providers sometimes sell smartphones



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 3.2: Permissions required to install DroidWatcher

with their own marketplace instead of the Google Play Store.

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 typically made using a USB cable).

3.3 Attack schemes

Even though the multiple installation procedures make the propagation of an application easier, they also enable abuses and accelerate the propagation of malware. Figure 3.3 presents the different possibilities for a malicious applications to propagate and be installed on a device. The attack types are described in Section 3.4.1.

3.3.1 Play Store publication policy

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

To distribute an application on the Android Play Store, the registered developer can upload their applications on Google Play servers and make it available in a few hours[6]. There is no human control before the publication of an application. The

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

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

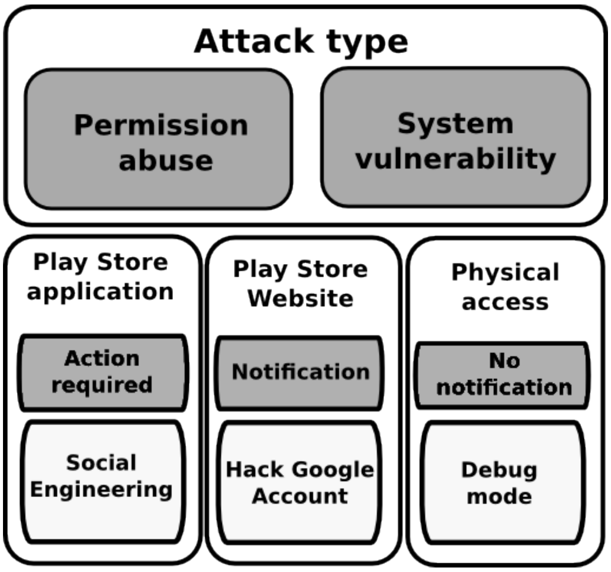


Figure 3.3: Malicious applications propagation possibilities

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 review the application which can take several days.

In February 2012, Google announced the creation of Bouncer, an automated scanning of the Android Play Store for potential malicious software. This scan is applied on every new and already published application to detect known malware, spyware and torjans or detected suspicious behaviour based on previous detection. This prevents detected malicious applications to be republished on different developer accounts or under different names [5]. However, in the 2012 DefCon conference, Bob Pan, from TrendMicro Inc., presented a way to bypass this security by dividing a known virus in several parts and recreating the infected apk afterwards [20].

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

This difference between Google and Apple regarding the verification process and ease to create a developer profile 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. Approval process to application publishing of the *Amazon AppStore*[8] developed by Amazon.com Inc. is similar

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

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

to Apple's regulation regarding iOS applications[9].

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

3.3.3 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. Declaring 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 assuming the Play Store is more secure than it actually is. This may lead to reduce the suspicion of the users and make the attack more effective.

Also, websites have appeared on the web providing 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 similar to the warez websites on the Windows operating system: highly risky in term of malware propagation.

3.3.4 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 a 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.

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

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

3.4 Malware

3.4.1 General malware types

There are two main types of Android malware: 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 permission allowing a malicious behaviour. This is usually the case with 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 without regard to the version of the operating system version that is being run. 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 grounds of the required permissions usually are not mentioned by the application makers[12].

The use of security flaws is possible due to the slow update process. The manufacturers tend 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 the users owning such devices is to install alternative ROMs such as CyanogenMod that provide an extended support for a large range of devices. When a security flaw is discovered and a patch published, only a very small percentage of users will benefit from the patch through an update in the months following the discovery. Malware writers can subsequently create programs taking advantage of that deficiency.

3.4.2 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) was used. Referring to Figure 3.3, it is classified as using social engineering to exploit a system vulnerability.

The attackers created several developer 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 up to the version 2.2 (99% of the devices at

[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`

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 speculated it could be monetised in the future to be used as a botnet (spam sending, distributed denial of services, etc.).

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

3.4.3 Protection

Observing the large increase of malware applications on the Android platform[16], users and developers wonder about the need of antivirus software. Similarly to the principles employed by antivirus for desktop computers, this antivirus software usually works using malware database basis. Such software would be efficient on antivirus using flaws and deriving in several applications similarly to the DroidDream malware. However, in case of the second kind of malicious applications, the efficiency of the antivirus is mitigated as it is very easy and quick to develop applications abusing the granted privileges.

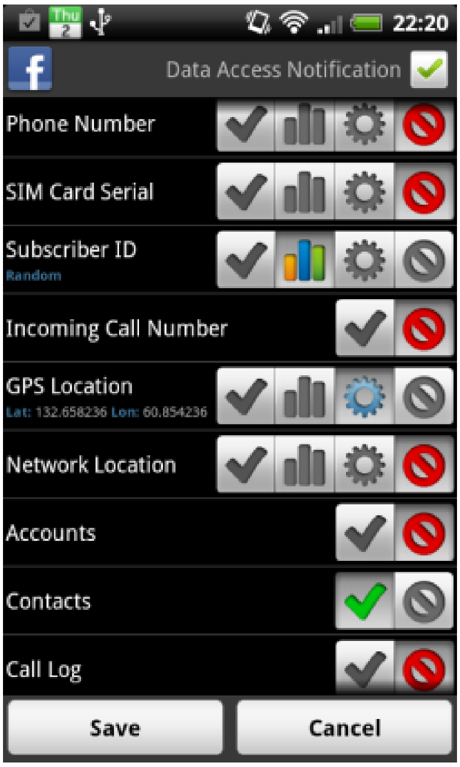
The PDroid application[17] takes a different approach than the antivirus software. Instead of detecting the known malicious applications, it allows the users to redefine the granted permissions and revoking them when desired. Another possibility instead of denying the access to certain information is 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 be useful to detect abuse of permissions. However, as the PDroid application works as an 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 most of the time are 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 substantial improvement of the permission model and we can hope a similar model to be adopted in future versions of Android. Figure 3.4 is an example of usage of the PDroid application capabilities on the Facebook application.

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

[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`



Data Access Notification

Phone Number

SIM Card Serial

Subscriber ID
Random

Incoming Call Number

GPS Location
Lat: 132.658236 Lon: 60.854236

Network Location

Accounts

Contacts

Call Log

Figure 3.4: PDroid on the Facebook application

3.5 Android ecosystem versus security

Section 3.3 and 3.4 explained the different possibilities of attacks as illustrated in Figure 3.3. These different types of attacks are partially related to the ecosystem of Android. The ecosystem of Android is very different from the one in iOS and this has an impact on the security of the platform.

The Android market is very fragmented, any manufacturer being able to port and deploy the Android operating system to its own smartphone models. When a manufacturer uses Android on a smartphone, Google loses the full and efficient control of the system. The system update process or even the presence of the Android Play Store is dependent of the commitment from the manufacturer.

This lack of a strict control requires the manufacturer's action to publish fix of known vulnerabilities and to manage third-party applications. The reaction delay will be unavoidably longer than if pushed by Google or even nonexistent to many devices (manufacturers preferring launching new models than supporting old ones). The availability of the code for porting and the current open nature of Android would make it very hard to gain more control in the future. The large choice of devices and installation medium (Play Store, alternative marketplaces,...) is a marketing argument but also a structural weakness.

Chapter 4

DroidWatcher

4.1 Presentation

DroidWatcher is an application developed for the purpose 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 actively monitoring the device movement has been developed. For the purpose of this thesis the application takes only acceptable actions for the respect of user privacy. But of course more invasive actions could be taken by malicious developers.

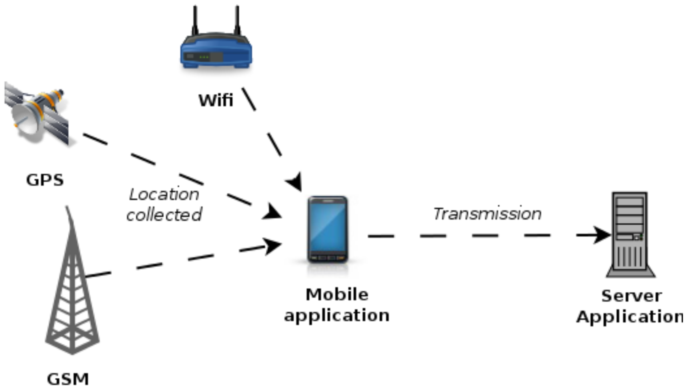


Figure 4.1: DroidWatcher system architecture

DroidWatcher is made of two major components as represented in Figure 4.1:

- 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 these smartphones have an Internet access.

In this chapter a detailed overview of the mobile application is given. 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 4.2.5

4.2 Mobile application execution

4.2.1 Application structure

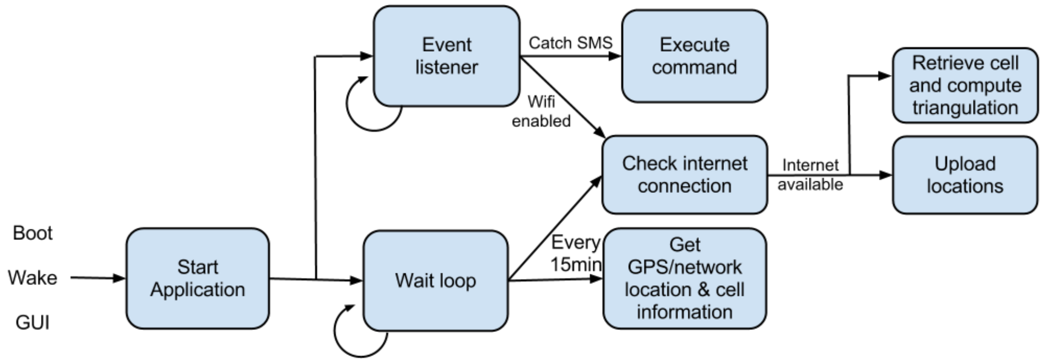


Figure 4.2: DroidWatcher execution process

Figure 4.2 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 launch two main components:

- Periodic actions
- Event listener

Every specified amount of time (15 min by default), the current location and surroundings cell towers information (identifier and power strength) are recorded. Is also verified if the phone is connected to the Internet to execute the online actions (upload of collected location and past cell phone towers localisation, see Section 4.2.3).

The event listener reacts to two types of event. In the case of an SMS received, it proceeds to the adequate action (see Section 4.2.4). The second watched event is the enabling of the wireless option on the mobile phone. This event will test if an internet connection is successfully established and execute the online action in case of success.

4.2.2 Location recording

At the installation, the mobile application requests the localisation permissions[3]. Using these permissions, every specified time interval (15 min by default), DroidWatcher records the location of the device. The application works in background and keeps recording locations when the user does not use the smartphone. Only the

[2] Except with Android 4.0 or above where the interface should be launched at least once, see Section 4.4.2 for technical information

[3] See Section 3.1 for details about the permission model

most accurate location is kept in an interval of 15 min. Both GPS and network[4] resources are monitored using the officialy provided methods in Android. If a new 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.

4.2.3 Internet actions

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

- Retrieving the geographical coordinates of the collected GSM cell towers and computing the previous locations using triangulation
- Synchronising 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 4.3.

4.2.4 SMS management

The mobile application provides a SMS-based command interface. These commands allow configuration of the application (frequency of retrieval, server url, etc.) and take direct actions related to the location (enable GPS, return current location, etc.). 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.

[4] The network resource is defined as the use of wireless and cell tower access points as described in Section 2.2 and 2.3

[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.

4.2.5 GPS activation

The GPS of a device can be remotely activated using 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, 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 a user to understand that such flaws exist 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).

4.3 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 connected to the Internet.

4.3.1 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 towers are 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 4.4.3.

4.3.2 Determine the position

Depending of these three scenarios, the respective centres will be decided as:

1. The location of the only GSM tower within range

2. Along the median between the two GSM towers

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

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.

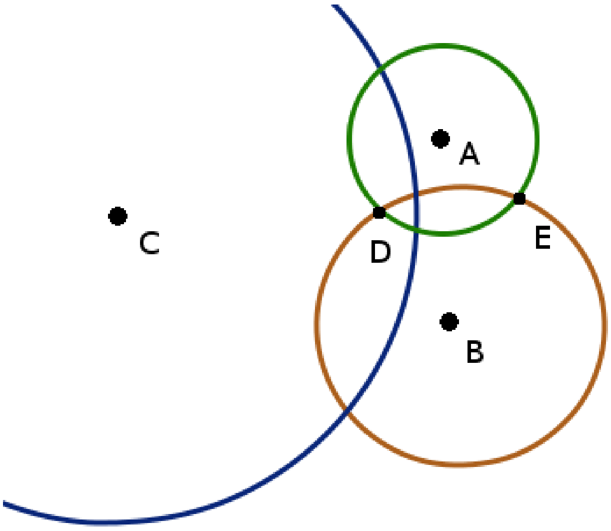


Figure 4.3: Triangulation scenario with 3 GSM towers

The third scenario is illustrated in Figure 4.3. The tower A and B are considered to be 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 metres
2. The distance between the two intersections
3. The half of the distance between the two intersections

4.3.3 Signal strength conversion

If a device receives 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 take. The stronger the signal is, the closer the device is 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 decibels[7]. 0 ASU representing the lowest signal and

[7] For the GSM network, $dBm = 2 * ASU - 113$

31 the strongest. The conversion is expressed by the following formula :

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

The weighting of the factors has been decided based on personal observations using the author's smartphone. Successive locations were recorded and compared to the location of the cell tower and signal strenght.

4.4 Technical challenges

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

4.4.1 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 during its 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 the position. The GPS may sometimes 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 from the application but is the direct consequence of battery saving measures taken by the system. This issue is often a complaint related to the tracking application (eg: sport monitoring application). A solution is stop the phone from 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 consumption of the phone and consequently the effectiveness of the application in monitoring the location for an extended period of time and as discretely as possible.

4.4.2 Android 4.0

The author of this thesis owns a device with Android 2.3 and this device was used in the process of the research. At the time of development, end of 2011, the fourth version of Android[9] 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 manages 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 supported the aim to be fully discrete and ensured that the application was not noticeable without analysis. With Android 4.0, an application can no longer

[8] See Section 2.1 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

start during the boot or after having been woken up if the interface has not been launched first time [11].

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 to run a malicious *invisible* application. However malicious applications often use a fake interface (weather forecast, game, etc.) to hide the malicious behaviour of the software and this protection will therefore not affect these applications.

4.4.3 Cell tower triangulation

To compute the location of the user, a triangulation algorithm has been developed. This algorithm uses a circle intersections [17] and a coordinates conversion [21] algorithms to compute the triangulated coordinates. However the unknown factor is the relation between the received signal strength and the distance from the cell tower.

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 of being able to record a reasonable approximation of the location at any time when GSM connectivity is available.

Part II

Wireless cameras

Chapter 5

Introduction to wireless cameras

Personal wireless surveillance cameras are widely available for general public nowadays. From hotel receptions to home surveillance or supervision of the babysitter[1], many reasons can justify the usage of a video camera. While these cameras have valid usages, it is an open door to very important privacy threats. The possibility for unauthorised people to access the video could have the opposite effect of protecting something (helping the work of potential burglars for example). During the placing of this kind of wireless cameras, the question of security and privacy is often neglected or completely forgotten.

This chapter describes the different types of cameras and possible abuses on the web.

5.1 Analogue cameras

Analogue video cameras are often cheaper than the digital models and easier to install (no wifi network required). This explains the fact they are still in use nowadays and may still be used in the future for shops and in private sector. The video camera records analogue pictures and transmits them on a defined radio frequency. To visualise the video, a receptor is set on the same frequency. The frequency bands used are usually around 2,4GHz and have been selected so to avoid conflict with other emissions.

The important aspect about these cameras is the fact that, as the video is broadcasted on a defined frequency, any receptor set on this frequency can receive and watch the video stream. This implies that technically there is no protection of the video stream. Anybody using a portable receptor is able to watch the video stream (which could also contain an audio stream depending on the camera model) at a range of few metres.

The usage of this type of camera should only be reserved for non sensitive content and be considered as available to anybody.

[1] Miniature wireless cameras called *nanny cam* are sold on specialised websites for child safety.

5.2 Digital cameras

Unlike the analogue video cameras, the digital models record digital images and transmit them using numerical stream, in this case, using wifi networks. The protection of the video stream is then possible at several levels:

- Encryption of the wireless network (WEP, WPA, WPA2, etc.)
- Encryption of the transmission protocol (usage of SSL to transmit the images)
- Access control mechanism of the stream itself (password to access the admin interface)

The security of the video stream is then directly related to the choice of the security mechanism. Using a weak security mechanism (for example WEP) leads to a weak security of the camera.

The digital video cameras can contain many features for the transmission of the video stream. As these features improve the access convenience using different methods (admin interface, mail sending, etc.), they also put a challenge on the side of the end user who must understand the key differences in terms of security. For instance setting an admin password to the administration interface during the configuration process does not protect the RTSP[2] stream in the analysed D-Link camera.

5.3 Web digital cameras

Some digital cameras are available on the Internet . This section explains how a camera can be accessed from outside the local network and what are the risks of discovery and abuses.

5.3.1 Connectivity

The default deployment of a digital camera is to be restricted to a home network using wifi. However, it is possible to make the video available on the Internet. This is done by configuring a port forwarding at router level. Then the public IP address of the network can be used to access the web interface of the camera. Some camera systems also allow streaming to external websites or implements dynamic DNS mechanisms to be able to access the video stream from an URL.

Another common usage of web digital cameras is for outdoor cameras streaming streets, parks, ski resorts and other public areas. Unlike previously mentioned personal cameras, these cameras may be published and advertised on city websites or neighbourhood associations.

5.3.2 Google indexing

If the URL to a web camera interface is published on a website (legitimate or not), it may be detected and indexed by a search bot software such as the ones Google

[2] RTSP, for *Real Time Streaming Protocol*, is a streaming protocol, often used to embed the video stream in third party softwares.

uses to constitute its search engine database. If indexed, these cameras can be listed with the correct search keywords.

Using the Google search engine, the keyword `inurl:` is used to search in the site's URL itself. On certain models of digital cameras, the URL to access the web interface contains some patterns typical for these cameras. This is the case for example with `axis-cgi/mjpg` or `ViewerFrame?Mode=` that are present mostly in webcam server softwares. Using the following search queries can list links to the web interface of digital cameras:

- `inurl:ViewerFrame?Mode=`
- `inurl:ViewerFrame?Mode=Refresh`
- `inurl:axis-cgi/jpg`
- `inurl:axis-cgi/mjpg`
- `inurl:view/indexFrame.shtml`
- `inurl:view/index.shtml`
- `inurl:view/view.shtml`

While these cameras are, most of the time, filming public areas were already available on websites without restrictions, these search queries allow quickly to list a large number of cameras. By refining the search with other keywords such as *airport* or the names of cities, it may make easier the collection and cross-checking information and facilitate abuses.

5.3.3 Detection of non-public cameras

To end up in Google results, all the cameras have been published on a web page that has been indexed by Google. However, it is possible to find camera streams of more private areas such as bars, factories or even homes. Most likely, the video stream from these cameras has not been published by their owner on a website that was indexed by a search engine bot. Often, the owners of personal cameras rely on the fact that the IP address or URL to access these cameras are not known and published. However it is possible to find and access the video stream of such cameras in some cases. Two main methods can be used to detect these private cameras:

If a dynamic DNS system is used (some cameras integrate DDNS clients in their configuration), the URL address can have a form similar to http://<*client-id*>.<*ddns-provider*>. This is the case for example with AXIS[3] surveillance cameras system which provides a dynamic DNS system URL with the form http://<*client-id*>.axiscam.net. By iterating or using a dictionary on common keywords, it is possible to find cameras that were not intended to be published.

Another method to detect cameras is to scan ranges of IPs addresses. By iterating on a range of IP addresses, it is possible, with a sufficient amount of time, to list any type of IP device, including digital cameras, accessible via a public IP address. Some cameras have unique patterns in the URL or in the header of the requests,

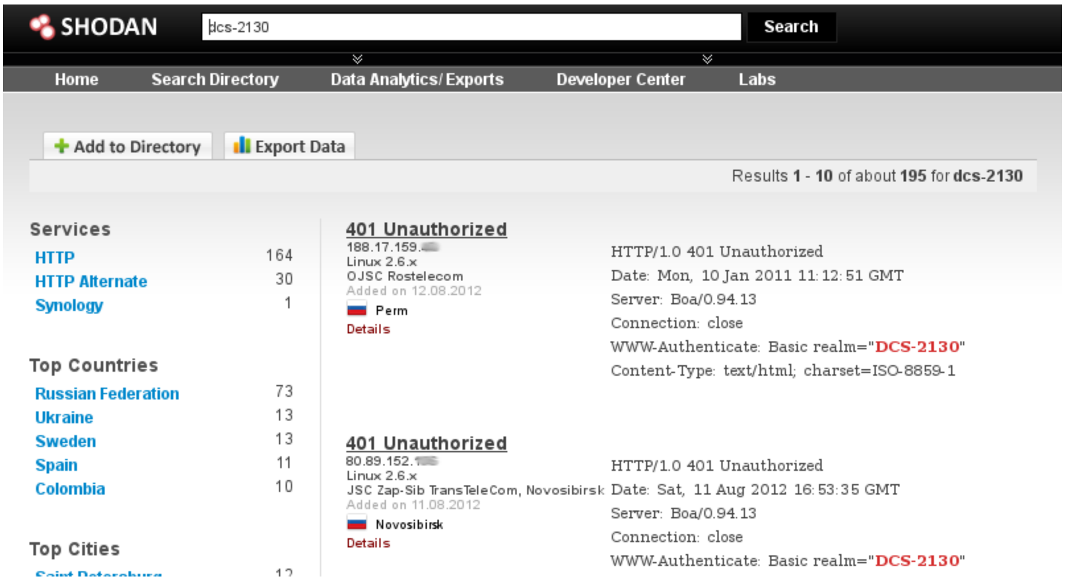
[3] AXIS `http://www.axis.com/`

searching for these patterns allows to detect these hidden cameras. A script detecting the model of D-Link cameras studied in Chapter 6 is described in Section 6.8.

5.3.4 Shodan

Shodan[4] is a search engine targeted at finding online devices. Instead of crawling the content of web pages, it scans IP ranges using headers, IP and ports to detect devices connected to the Internet.

In the case of camera, this search engine can be used to detect web cameras available on the Internet. In case the camera interface is not protected by an access control mechanism, it is possible to find watch the video stream with the help of Shodan. The usage of an access control mechanism does not prevent the indexing of a camera by Shodan but helps to avoid unauthorised accesses. Figure 5.1 shows an example of a search done to find D-Link DCS-2130 camera, finding 195 devices.



Results **1** - **10** of about **195** for **dcs-2130**

Services	
HTTP	164
HTTP Alternate	30
Synology	1

Top Countries	
Russian Federation	73
Ukraine	13
Sweden	13
Spain	11
Colombia	10

Top Cities	
Saint Petersburg	12

401 Unauthorized
188.17.159.[redacted]
Linux 2.6.x
OJSC Rostelecom
Added on 12.08.2012
Perm
Details

```
HTTP/1.0 401 Unauthorized
Date: Mon, 10 Jan 2011 11:12:51 GMT
Server: Boa/0.94.13
Connection: close
WWW-Authenticate: Basic realm="DCS-2130"
Content-Type: text/html; charset=ISO-8859-1
```

401 Unauthorized
80.89.152.[redacted]
Linux 2.6.x
JSC Zap-Sib TransTeleCom, Novosibirsk
Added on 11.08.2012
Novosibirsk
Details

```
HTTP/1.0 401 Unauthorized
Date: Sat, 11 Aug 2012 16:53:35 GMT
Server: Boa/0.94.13
Connection: close
WWW-Authenticate: Basic realm="DCS-2130"
```

Figure 5.1: Shodan search to find D-Link DCS-2130 camera

5.3.5 TRENDnet vulnerability

While the methods mentioned previously abuse from a lack of protection, some cameras, even protected, suffer from vulnerabilities and the video stream can be accessed if the camera is available on the web. This is the case of a set of TRENDnet IP cameras on which a bug has been discovered.

In January 2012, the author of the webblog Console Cowboys, has published several researches on a TRENDnet digital camera [1]. Through his researches, he discovered on the server, the presence of an executable binary file `mjpg.cgi` in a public cgi directory. When executed, this file generates a video stream with regular snapshots of the camera. By accessing the file through the URI

[4] SHODAN - Computer Search Engine `http://www.shodanhq.com/`

`/anony/mjpg.cgi`, any user, even unauthenticated, would receive the video stream from any TRENDnet camera using this firmware as the example in Figure 5.2 shows.

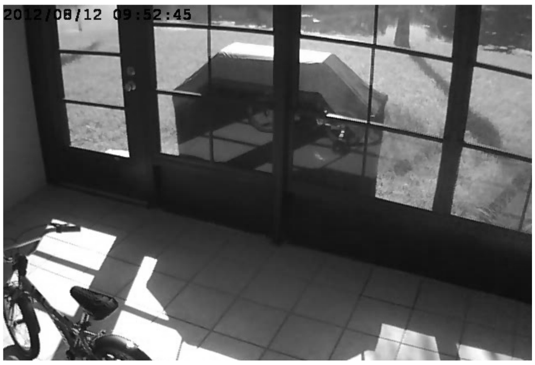


Figure 5.2: Example of video stream accessed on a vulnerable camera

The exploitation of this vulnerability is facilitated by the existence of the Shodan search engine presented in Section 5.3.4. Using the query `netcam` which is used in the headers of TRENDnet cameras, the Shodan search engine lists IP addresses of TRENDnet cameras. Using this list, the vulnerability can be tested on a large number of devices.

TRENDnet replied to this publication by proposing an update of the firmware that fixes the vulnerability and contacting registered owners to warm them about the potential threat. However it is very likely that most of the vulnerable devices will not be updated by their owners and still be exploitable in the future.

This vulnerability is an example of the danger of setting up such devices on the web. Even by protecting correctly and not publishing the URL to access the camera, owners of these devices are exposed to curious eyes as for the moment their camera is made accessible from the Internet. Unlike the results from the Google Search in Section 5.3, these cameras are often personal cameras and are often used in private areas. A similar bug permitting the access to the log file on the studied camera has been discovered and is explained in Section 6.7.

Chapter 6

Security analysis: case study of D-Link DCS-2130

Introduction

As part of this thesis, it has been decided to analyse the security aspects of one model of wireless digital camera, the D-Link DCS-2130. The camera has been selected as being recently released (2011), having good reviews, a large choice of features[1] and being at reasonable price range (around 100€). No previous security researches were found to be done on this model of camera. The aim was to select a common camera and take it as a representative example of the personal wireless camera market.

Figure 6.1: D-Link DCS-2130 camera

Some of the vulnerabilities and security aspects analysed in this chapter are specific to this model of camera. However as weaknesses have been discovered on other models of cameras (as showed the TRENDnet hack) it is believed similar issues can be discovered in other models.

6.1 DCS-2130 features

The selected camera model is a D-Link DCS-2130 and is represented in Figure 6.1. The camera supports the following protocols:

[1] Technical details about the device can be found on D-Link website at `http://www.dlink.com/`

- Wifi: Open, WEP, WPA, WPA2
- Admin interface: HTTP, HTTPS
- Access: UPnP, DDNS
- Transfer: FTP, SMTP, Samba, RTSP
- Video: H.264, MPEG-4, MJPEG
- Audio: G.726

The DCS-2130 has also features such as motion detection allowing to program conditional chain of events (eg: if a motion is detected, upload current pictures to a FTP server). The camera is provided with an installation utility to configure it from a computer to be able to connect to the wireless network. Watching the video stream and configuration of the system can be done using a web interface.

6.2 Firmware source code

As many embedded devices, D-Link uses a Linux kernel and open source software such as Busybox which are released under a GPL licence. A clause in this licence requires to make available the source code of the software using code under GPL if binaries are proposed (which is the case of D-Link). In 2006, the Court of Frankfurt issued that D-Link was violating the GPL licence on a network-attached storage device [6]. Following that ruling, D-Link published the source code of several of its devices on their website[2] including the DCS-2130.

Even if D-Link technically provides the source code of the software on their website, many difficulties have been faced to obtain the archive file. The difficulties were:

- No direct link to the file download (impossible to restart an interrupted download)
- Download speed highly throttle (from 1 to 60Kb/s max.)
- Regular connection timed out
- The contact email address provided for this section of the website is invalid
- No answer while using the general contact form

Given the nature of the difficulties, one can question if they are intentional to avoid the spreading of the source code too easily. The source code has finally been retrieved[3] and the analysis permitted a better understanding of the firmware mechanism as well as the discovery of security issues.

6.3 Installation procedure

6.3.1 Admin account

The installation and configuration of the software are realised using an ActiveX wizard utility provided with a CD-ROM included with the camera. This utility is

[2] D-Link GPL Source Code Support `http://tsd.dlink.com.tw/GPL.asp`

[3] The archive downloaded is available on a public FTP server at `ftp://ftp.dotzero.me/DLink/`

used to configure the connectivity aspect of the camera and to secure the admin account. The administrator account is used to manage the configuration settings of the camera through a the web interface.

This last point is positive as the procedure does not permit the configuration of the camera without setting a control access mechanism. However, this control access mechanism is not strong enough to prevent unauthorised visioning as detailed in Section 6.5.

6.3.2 Clear-text authentication

At the launch of this wizard, messages broadcasted are exchanged to discover the presence of compatible cameras. These messages use a proprietary protocol but understandable to anybody listening to the traffic. The problem of this protocol is the fact that, in the case the camera already is configured for the first time, the authentication of the admin account is required and this communication is also broadcasted, communicating the administrator password in clear text over the network.

Below are parts of an exchange between a laptop using the utility and a the camera.

```
   Laptop
0000  ff ff ff ff ff ff 08 00  27 07 34 7d 08 00 45 00   ........ '.4}..E.
0010  00 32 00 33 00 00 80 11  78 d6 c0 a8 01 0a ff ff   .2.3.... x.......
0020  ff ff 04 04 f6 00 00 1e  a2 fc fd fd 01 00 a1 00   ........ ........
0030  ff ff ff ff ff ff 00 00  00 00 00 00 01 00 00 00   ........ ........

   Camera
...
0080  44 4c 69 6e 6b 43 61 6d  00 00 00 00 00 00 00 00   DLinkCam ........
...
00c0  44 43 53 2d 32 31 33 30  00 00 00 00 00 00 00 00   DCS-2130 ........
00d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00e0  31 2e 32 36 00 00 00 00  00 00 00 00 00 00 00 00   1.26.... ........
00f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0100  01 00 27 00 00 00 f0 7d  68 09 52 52 44 4c 69 6e   ..'....} h.RRDLin
0110  6b 43 61 6d 00 00 00 00  00 00 00 00 00 00 00 00   kCam.... ........
...
0140  00 00 00 00 00 00 00 00  00 00 00 00 c0 a8 01 07   ........ ........
0150  50 00 02 00 ff ff ff 00  c0 a8 01 01 c0 a8 01 01   P....... ........
0160  00 00 00 00 01 32 30 31  32 30 36 32 38 30 36 35   .....201 20628065
0170  31 34 34 00 00 42 00 ff                            144..B..


   Laptop (after giving the credentials to the software)
0000  ff ff ff ff ff ff 08 00  27 07 34 7d 08 00 45 00   ........ '.4}..E.
0010  00 b2 00 34 00 00 80 11  78 55 c0 a8 01 0a ff ff   ...4.... xU......
0020  ff ff 04 05 f6 00 00 9e  82 f7 fd fd 02 00 a3 00   ........ ........
0030  f0 7d 68 09 52 52 c0 a8  01 07 77 77 01 00 80 00   .}h.RR.. ..ww....
0040  59 57 52 74 61 57 34 3d  00 00 00 00 00 00 00 00   YWRtaW4= ........
...
0080  62 58 6c 77 59 58 4e 7a  00 00 00 00 00 00 00 00   bXlwYXNz ........
...
```

Where `YWRtaW4=` is `admin` in base64 encoding and `bXlwYXNz` is `mypass`. These were the username and the password of the camera at the time of the experiment. While the meaning of each byte would require a larger study, it is however easy to guess the structure of the communication based on the observation:

1. Laptop: Broadcast detection message
2. Camera: Broadcast presence, model and configuration state
3. The software asks for the administration credentials
4. Laptop: Broadcast the credentials

The fact that all messages are broadcasted implies that, even on encrypted network using session keys such as WPA (which prevents monitoring the traffic of other users on a wireless network), the credentials are publicly transmitted. An attacker only needs to monitor the network to be able to retrieve an administrator access to the camera. However, the usage of the utility to reconfigure the network is supposed to be very rare and minimise the risk of such attack.

6.4 Security against traffic monitoring

As mentioned in Section 6.1, the camera supports several protocols. These protocols are not all as secure and while only monitoring the network an attacker could retrieve sensible information.

The monitoring of a network using free tools such as *tcpdump* is possible while an attacker is connected to the same network as the camera and when the wireless network traffic is not encrypted or using WEP[4]. On encrypted traffic, an attacker could apply an ARP spoofing attack to force the traffic to transit by its computer and read the content of packages.

6.4.1 HTTP authentication

The access to the web interface allows to watch the video stream but also configure the camera. This page is protected with a basic authentication mechanism over HTTP. This authentication is not secured as the credentials are sent in clear text in the headers of the request. The code below is extracted from the headers of a request to the web interface which contain `admin:mypass` in base64 encoding.

```
GET / HTTP/1.1
Authorization: Basic YWRtaW46bXlwYXNz
```

6.4.2 URL parameters

To configure the access to external services (email, FTP, DDNS, etc.) the administrator uses the web interface. This interface is accessed by default using HTTP which means that the URL of the requested pages is in clear text on the network.

For each configuration request, the URL in the headers contains the information for the selected service. For example, a request to update the password of the wireless access point credential would look like:

```
GET /cgi-bin/wifi_config.cgi?enable=1&ssid=MyWifiNetwork&Mode=0&\
   auth=3&encrypt=2&wpa_key=MyWifiPassword HTTP/1.1
```

This means that, even if other protocols are secured (eg: usage of SSL in SMTP), the configuration may be the source of information leakages.

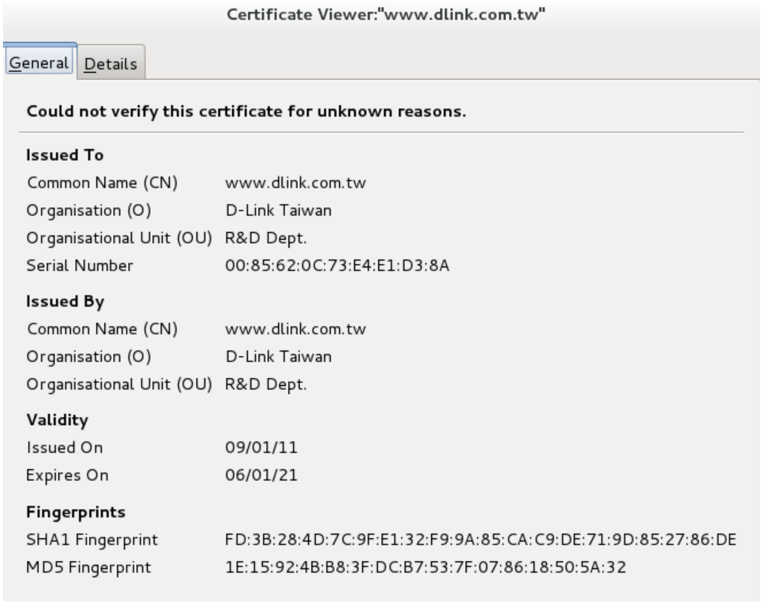
[4] Unlike WPA/WPA2, WEP does not use unique session keys for each user.

6.4.3 Export configuration

The camera features a capability to backup the configuration of the camera, by exporting current settings and information stored in the camera. In the administration panel on the web interface, the functionality *Save configuration* generates a text file containing all the specified information. This file contains, in clear text, all the passwords and login information specified during the configuration (user accounts, wireless access, mail credentials, etc.).

Apart from the fact that it is worrying that all this information is unnecessarily stored in clear text instead of an encrypted version, this also means that monitoring the traffic while a backup is done may reveal the content of the file. As for the URL parameters, using secured transmission protocols can be useless if the credentials can be retrieved while monitoring the network.

6.4.4 SSL certificate as a solution



Certificate Viewer:"www.dlink.com.tw"

Could not verify this certificate for unknown reasons.

Issued To	
Common Name (CN)	www.dlink.com.tw
Organisation (O)	D-Link Taiwan
Organisational Unit (OU)	R&D Dept.
Serial Number	00:85:62:0C:73:E4:E1:D3:8A
Issued By	
Common Name (CN)	www.dlink.com.tw
Organisation (O)	D-Link Taiwan
Organisational Unit (OU)	R&D Dept.
Validity	
Issued On	09/01/11
Expires On	06/01/21
Fingerprints	
SHA1 Fingerprint	FD:3B:28:4D:7C:9F:E1:32:F9:9A:85:CA:C9:DE:71:9D:85:27:86:DE
MD5 Fingerprint	1E:15:92:4B:B8:3F:DC:B7:53:7F:07:86:18:50:5A:32

Figure 6.2: Generated self-signed SSL certificate

To prevent the monitoring issues while using the web interface, it is possible to use a SSL certificate to establish a secured HTTPS connection to access the web interface. However, the certificate used is either self-signed and generated by the camera or uploaded by the user. Few people will configure an SSL access to the camera but even fewer will have a valid SSL certificate to upload. It is then very likely that most of the concerned users will use a generated self-signed certificate.

In Figure 6.2, the imported self-signed generated certificate for the case study camera is displayed in Firefox with a security warning. This means that an attacker could issue a forged certificate pretending to be the D-Link authority without the possibility for the end user to verify the validity.

6.5 Guest account

During the analysis of the camera, it was revealed that the exported configuration file mentioned in Section 6.4.3 contains also the information about the different users allowed to access the camera with their access right level as shown below.

```
acounts0_name=admin
acounts0_passwd=mypass
acounts0_authority=0
acounts1_name=guest
acounts1_passwd=guest
acounts1_authority=2
```

In this part, an account *guest* using the password *guest* appears with an access level of 2. This account is automatically created at the installation of the camera.

It is important to notice that neither during the installation process nor in the user manual, the existence of a guest account is mentioned. The only way to detect its presence is to analyse the exported configuration file or see it in the list of removable accounts in the configuration interface (but without knowing the password). While the obligation to set up a password for the administrator account is positive in terms of security, the hidden creation of the guest account pushes the end user in an incorrect sense of security.

6.5.1 Guest account access

The guest account is configured with an access level of two, this level prevents the guest user to modify the configuration of the system but is enough to both watching the video stream and retrieving snapshots of the video using a specific URL.

In the firmware source code, the web interface is powered by the open source web server Boa[5]. To resolve the different URLs, this webserver lists all the possible URI with the associated function to execute and the access rights required to access the URI[6]. Using this list, it is possible to identify what the guest account is capable of. Fortunately, the guest account appears to be able only to access the stream of the camera and some moderately important information such as the firmware version.

6.5.2 Account suppression

The presence of the guest account is very problematic if a user wants to configure the camera as visible from outside (see Section 6.8). In the administration web interface, it is possible to create and delete users. However, the removal of the guest account has been specifically prevented. In the webserver source code, the case of removing the guest user is refused as shown in the code below present in the function responsible for removing an account[7]

```
...
#ifdef CONFIG_BRAND_DLINK
    else if(i == 1){
       DBG("You can not delete the guest accunt!\n");
       break;
```

[5] Boa Webserver `http://www.boa.org/`

[6] In the firmware source code, see `/apps/public/boa-0.94.13/src/request.c` line 6700

[7] In the firmware source code, see `/apps/public/boa-0.94.13/src/request.c` line 3280

```
    }
#endif
...
```

This piece of code (typographical error in the original code) is highly surprising as a clear addition from D-Link as an effort to prevent the removal of the guest account (identified by the variable `i == 1` here). The reason of this effort is unknown and left to the reader interpretation but it leads to an important security breach as it facilitates the unauthorised viewing of the video stream.

As the modification or removal of the guest user is not possible using the web interface, another procedure as been discovered:

1. Export the configuration file
2. Using a text editor, empty or modify the lines related to the guest account
3. Import the modified file as the new configuration

Emptying the lines will delete the guest account while modifying can keep a guest account present but with a different password. This procedure is the only known solution to prevent unauthorised access to the web interface using this account.

6.6 RTSP protocol

The RTSP protocol (for *Real Time Streaming Protocol*) is enabled by default. As mentioned earlier, this protocol is used to stream the video into another software. The video stream can be accessed using any compatible third party software using an URL in the form rtsp://<*camera-address*>/<*rtsp-path*>. Through the admin interface, it is possible to disable, change the port number or rename the access path.

Even if technically possible[8], the system does not implement any access control mechanism. If an attacker has access to the associated port (554 by default), it is possible to stream the video, even without necessity to use to the guest account.

If not used, it is recommended to disable this protocol.

6.7 Log file

As mentioned in Section 6.5.1, the allowed queries are hard-coded in a list in the file `request.c` of the webserver. For some of the requests, there is no execution function associated, only access rights. This is often the case with requests to the folder `/cgi-bin/` which contains binary files to execute instead of applying a defined function.

During the analysis of the source code firmware, it was noticed that the binary file `/cgi-bin/exportlog.cgi` was present in the folder of the server but not in the URI list. This lack of definition in the request file implies that it will be executed

[8] The RTSP protocol can implement a basic access control such as it is done on HTTP to access the web interface. However it is not possible to enable this feature on this model of camera.

according to the file execution rights and not the permissions that could have been defined in the server code. This means that, even for an anonymous visitor (not guest or admin), it is possible to execute this file.

As its name implies, the file exports the log of the camera in a text file. This log file does not reveal highly sensitive information (no password) but gives the state of camera through time.

```
2012-05-24 21:01:29 NETWORK LOST
2012-05-24 21:01:29 SD CARD SIZE 7620040 KB
2012-05-24 21:01:34 NETWORK RECONNECT
2012-05-24 21:03:15 admin LOGIN OK FROM 193.44.55.11
2012-05-25 15:40:38 IP CAMERA Received MOTION Trigger
2012-05-25 15:40:41 MOTION STOPPED
```

A more sensitive information displayed in the log file is the login account name, the IP addresses and events such as motion detection activation. Analysing this information could allow an attacker to retrieve information such as the user habits and presence time.

Like the TRENDnet vulnerability showed in Section 5.3.5, the possibility to access the log file is due to a bug and it is not possible for the end user to prevent this security leak. The only possible mitigation is a regular cleanup of the log file in the administration interface (which few users would normally do).

6.8 Web camera discovery

As mentioned in Section 5.3.1, it is possible to configure a network to access a camera from outside of the network. However, this access may create the possibility for attacker to access the camera. Using the previously mentioned vulnerabilities, intrusions are possible to login as guest account or to access the log file.

As mentioned in Section 5.3.3, it is possible to scan a network in order to discover hidden IP cameras. While making a request to an IP address, the headers of the reply are specific to the IP device and make it possible to easily identify the presence of cameras. The reply headers of a request shown below contains the identifier `Basic realm="DCS-2130"` which can be used.

```
HTTP/1.0 401 Unauthorized
Date: Sat, 12 Feb 2011 17:59:32 GMT
Server: Boa/0.94.13
Connection: close
WWW-Authenticate: Basic realm="DCS-2130"
Content-Type: text/html; charset=ISO-8859-1
```

As a proof of contest, the program `dcs_detection.py` was developed. This script scans a given range of IP addresses until it detects a basic realm containing `DCS-2130`. In a few minutes, this script can discover the presence of a camera on a home network or on a public IP range. The example below shows the execution of the script detected two cameras in about 4 minutes on a /24 range of IPs.

```
$ time python dcs-detection.py 78.37.191.x
Iterating on 256 urls
```

```
78.37.191.73     --> DCS-2130 camera!
78.37.191.117    --> DCS-2130 camera!

real    3m57.994s
user    0m2.007s
sys     0m0.127s
```

Using Shodan search engine (discussed in Section 5.3.4) with the keyword `DCS-2130` is another efficient way to retrieve quickly a list of IPs connected to this model of camera. Using the developer API, the script `shodan-dcs.py` has been developed to list the indexed DCS-2130 cameras.

Using the list of discovered IPs can be exploited to access video streams as shown in Figure 6.3. Through personal experience, only one camera accessible from the web had the guest account not available but had the RTSP port open. Bottom line, no DCS-2130 camera found on the web was effectively able to prevent access to the video stream. This is the consequence of combination of the unknown presence of the guest account and the ease to detect such devices.

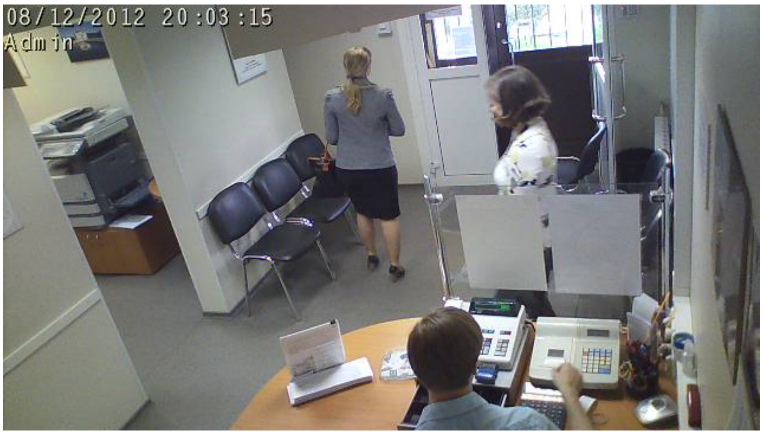


Figure 6.3: Snapshot from a DCS-2130 discovered while scanning an IP range

6.9 PRNG testing

Random numbers are used in many security protocols. The unpredictability of these random numbers are essential to ensure the secrecy while generating keys or initialisation vectors. The generation of true random numbers is a common challenge in computer sciences as a computer is deterministic. Instead, devices use arbitrary inputs such as disk moves or network noise to generate pseudo-random numbers. However, on small embedded devices such as wireless cameras, there is not as many possible inputs as on a personal computer (no hard drive for instance). A weak random generation procedure may lead to a weak usage of theoretically secured protocols.

As a way to evaluate the security of the camera, the quality of the PRNG (Pseudo-Random Number Generator) used by the camera has been verified. To evaluate the randomness quality of generated numbers, the tests from the NIST were used[9]. These tests are a recognised standard and allow to give a neutral evaluation of the security of the PRNG in the camera.

To retrieve as many bits of entropy as possible, the program `gen_https.py` has been developed.This script generates a specified number of SSL connections to the web interface of the camera. While this script was running, the network was monitored using the tool *tcpdump* recording the communication in the PCAP format. In each HTTPS connection, each participant of the connection uses a 28 bytes payload of random bytes. These bytes were extracted from the trace file using another developed script `pcap_to_random.py`. For each set of requests, the random bytes from the camera and from the laptop used in the experience were separated and evaluated using the NIST tests set.

The following methodology was used through the experiment

1. Start the *tcpdump* utility
2. Run the `gen_https.py` with 5000 connections
3. Stop *tcpdump*
4. Extract the bytes of entropy from the camera and the laptop using `pcap_to_random.py`
5. Run the NIST tests on both extracted random sequences and compare the results

The experiment was repeated several times in a row to test the evolution as the pool of entropy of both devices was emptied. As a set of 5000 connections uses 140Kb of entropy, experiment showed that one run is enough to empty the main pool of randomness on the test laptop.

The conclusion of these tests was that, in every tested run, the entropy of both the laptop and the camera were sufficient to pass the NIST tests. If the tests carried out on the laptop showed better results than the ones using the camera, both were still enough to be considered as random. The difference between the two devices can be explained by the fact that the camera has no hard drive and more external sources of entropy, allowing it to refill the pool of entropy faster than the camera.

6.10 Burglar scenario

As an example of issues related to lack of security from a wireless camera, the scenario of a burglar (with some computer knowledge) planning to visit a building equipped with a DCS-2130 camera is detailed. This is one possible abuse showing concretely the danger of forgetting the privacy aspect while using this device. Another possible abuse for professionals would be industrial espionage. Two scenarios are discussed in detail below depending of the situation of the burglar. Not every case may lead to an abuse of the camera.

[9] The NIST, *National Institute of Standards and Technology*, provides open source software available at `http://csrc.nist.gov/groups/ST/toolkit/rng/index.html`.

6.10.1 Physically close to the camera

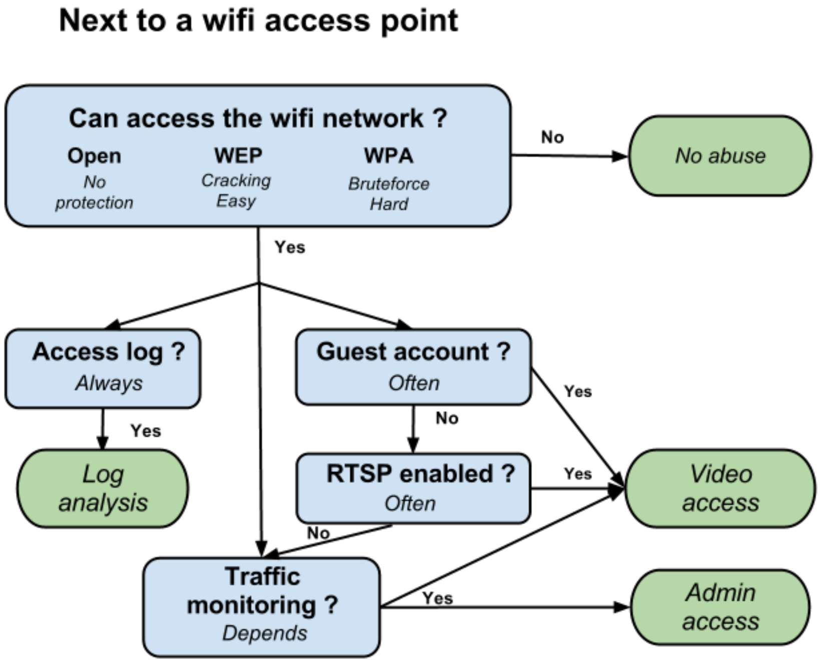


Figure 6.4: Case where the burglar is next to the access point

This first scenario represented in Figure 6.4 is the case where the burglar is in the range of the wireless camera while planning the visit. To be able to abuse the camera, it first requires to penetrate the wireless network. To achieve this goal, the protection of the network is directly related to the feasibility of this step. In the case of open network or WEP network this is possible easily but if WPA or WPA2 is used, this may be more difficult[10]

Once an access to the network is realised, the burglar has three main sources of information:

1. Access the log to retrieve information such as presence hour of inhabitants, motion detection system, etc. (always possible)

2. Access the video stream using RTSP or the guest account to spy the inhabitants of the building (often possible)

3. Gain an administrator access by monitoring the network for a longer period and abusing from the clear-text messages to retrieve credentials (depending of the possibility to monitor the traffic for an extended period of time)

Gaining an administrator access could be useful for the burglar to be able to disable camera surveillance mechanisms such as motion detection during the future visit. The access to the video stream also helps the burglar to detect the precise emplacement of the camera and avoid getting caught on tape during his misdeed.

[10] The wifi penetration techniques are not detailed in this section and left to the reader researches.

6.10.2 Web access to the camera

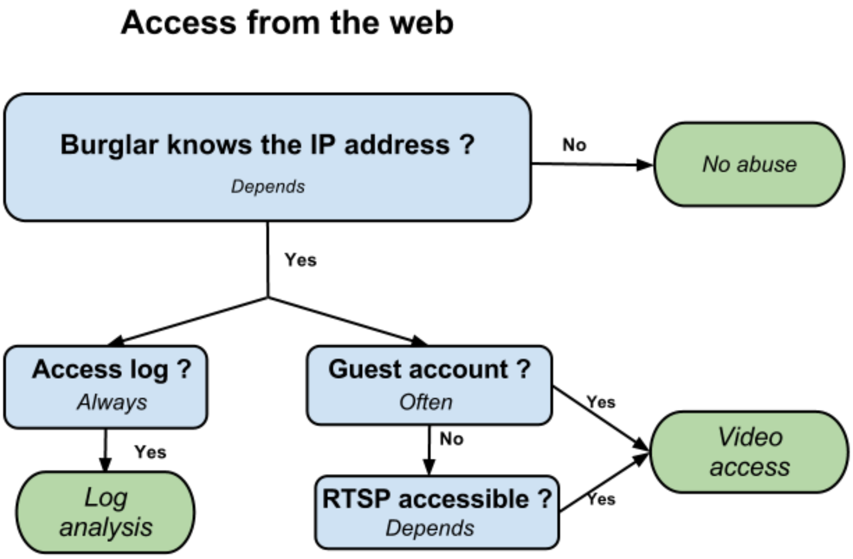


Figure 6.5: Case where the burglar access the camera from the web

The second abuse case represented in Figure 6.5 is when the camera is accessible from the web using the public IP address. The requirements are harder to achieve than the previous abuse case as the burglar needs to know the IP address of the targeted house (no other efficient techniques than scanning large IP ranges). Another possible scenario would be the burglar deciding the visit of a building based on the availability of a camera and obtaining the location based on the observations and IP address.

If the requirements are met, two possibilities of abuse are now possible:

1. Access the log to retrieve information such as presence hour of inhabitants, motion detection system, etc. (always possible)

2. Access the video stream using RTSP or the guest account to spy the inhabitants of the building (often possible)

The RTSP availability is less likely to be possible than in the first case as it requires a full access to the ports of the camera and not a simple port forwarding of the needed port (80 or 443 to access the web interface). However, during the experiments, some cameras where discovered as fully available from the Internet with the RTSP port open.

6.11 Security advises

To conclude the camera analysis, the usage of the camera without taking security measures is not considered as safe and is not recommended. The owners of this model of camera should apply the following steps:

- Disable the guest account.
- If not used disable the RTSP stream.
- Secure efficiently the wifi network (WPA or WPA2 with a hard to guess password).
- Use the web interface in HTTPS after making sure to import the certificate while in a secured environment[11].
- Cleanup regularly the cache file.
- Avoid configuring the camera when not necessary, especially using the installation utility.

6.12 D-Link reactions

Before the publication of this thesis, it has been tried to contact D-Link. Both the contact email address of the Benelux support and the community forum were used to try to establish a discussion with a D-Link member regarding these concerns. Unfortunately none of the attempted communications have led to an answer.

As it is possible the messages did not reach the adequate D-Link members, it is disappointing that it was not possible to contact them before the publication of this thesis. However, it is hoped that these issues will be noticed by D-Link and be fixed in future version of the firmware.

6.13 Future research

During the study of the camera, several researches where started but did not reach a conclusion and could be continued in future works.

6.13.1 Firmware modification

As the source code of the firmware was provided, it has been possible to build the source code into an installable image. Future researches could try to modify the firmware to develop a more hacker-friendly environment and being able to run deeper analysis of the behaviour of the camera while running.

6.13.2 Stream interception

Experiences showed the video traffic is accessible using an ARP spoofing attack. However, no attack on the traffic has been applied. A possible attack would be the on-the-fly modification of the video stream while using the RTSP protocol.

If technically possible, this attack is difficult to set up due to the complexity and variety of the different video and audio protocols. A similar attack on Cisco IP surveillance camera has already been demonstrated [15], showing the feasibility of such attack.

[11] Wired connection, no potential attacker on the network...

6.13.3 Port 1010

While scanning the opened port of the device, every opened port had a clear purpose (HTTP, RTSP, UPnP, etc.) with the exception of the port 1010. This port is opened but during the tests it was found that this port was never used, nor did it reply to tried queries.

A future research could determine what is the usage of this port and his eventual interest in the security analysis of the camera.

6.13.4 Other models of cameras

The discussed issues have been detected for the selected camera. As the time and resources were limited, only the DCS-2130 model was studied. However it could be interesting to extend the researches to see if the issues mentioned also apply to other models of camera.

Conclusion

Chapter 7

Thesis conclusion

Through this thesis, several points have been raised. The focus on the two case studies are believed to be representative of the current privacy issues.

7.1 Android ecosystem has to evolve

The Android system is not technically unsecured with the exception of some unavoidable security flaws. However the security issues are present and explained by two factors.

One of them is the negligence or ignorance of the security risks during the installation process, users will install malicious applications that could have been avoided. Applications similar to DroidWatcher are technically possible but the risk of propagation could be easily minimised with a better security awareness.

However, the user is not the only one responsible for security issues. In comparison to the model used in iOS, the open politic of Android is also responsible for the higher risk. An open politic has indubitable advantages for the user freedom but enables abuses and reduces the marge of action of Google (release of updates, use of Play Store, etc.) left to manufacturer responsibility. The permission system is also not perfect, it lacks from precision, clarity and enables too easily abusive behaviours.

It is questionable if Android will be able to react to the recent strong raise of malicious applications. Will antivirus softwares be mandatory as on the Windows platform or will a closed and certified marketplace such as the App Store be developed? It is however imaginable that an evolution in both the user behaviour and the platform security measures will happen to adapt to this risk. The fragmentation of the market should also be resolved by a better reaction to security threads to manufacturer (better support of older models, etc.).

7.2 Wireless camera must be shipped secured

The analysis of the DCS-2130 reveals several security issues that can be exploited. The usage of such camera without taking additional security measures is considered as unsecured and unadvised. It is possible to use the camera in a secure way but it requires a knowledge of security principles that most of the users do not have.

These security issues are not impossible to fix but it requires the manufacturer to dedicate more attention into security. The security of appliances such as a wireless camera need be present *out-of-the-box*, not requiring additional effort from the end-user.

7.3 Privacy does matter

On both Android systems and the wireless cameras, the potential leak of information has very dangerous consequences. A smartphone is able to track easily every movement of a user or reveal sensible private information. The case of camera abuses is easily imaginable as harmful as the burglar scenario showed.

The privacy is an important aspect of the usage of technology. If the state of care for privacy does not improve, scandals such as the one related to the cache database in iPhone and Android systems or the TRENDnet hack will happen again.

7.4 Beware of the convenience

The common aspects of these technologies is the convenience that drives their adoption. As these technologies are used as a service, they are considered only for the task they achieve and not as the system they are. The tendency of today is to abstract all the complexity from the user and sell a device that *just works*. However this is often untrue, a system is more complex than it is advertised. Security and privacy are some of the limitations of this abstraction.

7.5 Personal thoughts

I found the researches and analysis made during this thesis very interesting. The redaction of this document helped me to develop a critical view of the technologies we use every day. The analysed case studies reveals various security issues and enable potential abuses in terms of privacy that I did not suspect. I hope that reading this thesis will create a better awareness of the fact that these technologies are not perfect and that relying fully on them is not without risk. Fixing the security and privacy issues is not possible in one day, it starts with a better concern in these issues from the user but also better regulations protecting more the privacy of users.

Appendix

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 CD-ROM[1] to the Android device
2. Open the file with the Android installer
3. Accept 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 4.4.2, 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.

`BIGBRO` : starting code for a command.

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

- `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

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 starts at the phone boot and when the user unlocks 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.

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

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

8. Restart the apache module

   ```
   $ sudo service apache2 restart
   ```

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

Appendix B

Softwares

For the realisation of this thesis, several softwares have been developed and used for the researches. This appendix gives a short summary of the usage and purpose of the software.

The code of every software mentioned here can be found on `https://gitorious.org/martin-trigaux-thesis/code/`.

B.1 androdump.py

Download the content of the file system on the Android device. The ADB utility should be installed and the device connected to the computer making the dump. A root access is required to be able to retrieve the full content of the system. This program is used in experiments in Section 2.5 to monitor impact of actions on the file system.

```
Python 3
Usage: androdump.py [options]

Options:
  -v, --verbose         Enable verbose mode, lots of text
  -p PATH, --path=PATH  Starting path, default '/'
  -o OUT, --out=OUT     Outgoing path, default current path
```

B.2 compare_dump.py

Compare the content of two folders. The hash function `sha256` is used to compare two files and detect the changes. The script list the files that have changed were added or deleted between the two folders. This program is used in experiments in Section 2.5 to compare two dumps of the system made with `androdump.py`.

```
Python 3
Usage: compare_dump.py [options]

Options:
  -v, --verbose  Enable verbose mode, lots of text
  --path1=PATH1  Starting path for folder n°1
  --path2=PATH2  Starting path for folder n°2
```

B.3 LocateMe.apk

LocateMe is an Android application. At the launch of the application, it will display current wireless information such as the number of surround wireless access points and GSM cell towers. It will also make a single localisation request using the wireless network. This application was used for experiments in Section 2.5 as a simple way to create a localisation request and observe the according changes.

B.4 gen_https.py

Generate a specific number of HTTPS connections to a specified host. This program is used to generate a large number of bytes of entropy to test the quality of the PRNG as explained in Section 6.9.

```
Python 3
usage: Generate HTTPS connexions [-h] [--host HOST] N

positional arguments:
  N            the number of connexions to generate

optional arguments:
  --host HOST  IP address of the host to contact
```

B.5 pcap_to_random.py

Retrieve the 28 random bytes from the *server hello* packet during the initialisation of an HTTPS connection. It uses a PCAP trace file and outputs an ASCII sequence of bits. This program is used to extract a large quantity of random bits from a trace file to test the quality of the PRNG as explained in Section 6.9.

```
Python 2
usage: pcap_to_random.py [-h] [-s SOURCE] [-d] filename

A pcap random number extractor

positional arguments:
  filename              The pcap file to parse

optional arguments:
  -s SOURCE, --source SOURCE
                        The packet author address in the form
                        00:11:22:33:44:55
```

B.6 dcs-detection.py

This program scans a range of IP to detect DLink DCS-2130 cameras available on that range. The use of this script is explained in Section 6.8.

```
Python 3
usage: dcs-detection.py [-h] [-v] range

Scan a range of IP address to detect DCS-2130 cameras

positional arguments:
  range          The range to scan in the form 1.2.3.x

optional arguments:
  -v, --verbose  Verbose mode
```

B.7 shodan-dcs.py

This program uses the Shodan search engine presented in Section 5.3.4 to retrieve DLink DCS-2130 cameras indexed. An API key should be created at `http://shodanhq.com` and used for the variable `SHODAN_API_KEY`. The use of this script is explained in Section 6.8.

```
Python 3
usage: shodan-dcs.py
```

Bibliography

[1] Anonymous. Trendnet cameras - i always feel like somebody's watching me. http://console-cowboys.blogspot.be/2012/01/trendnet-cameras-i-always-feel-like.html, Jan 2012.

[2] Alan Davidson. Testimony of alan davidson, director of public policy at google. http://www.judiciary.senate.gov/pdf/11-5-10%20Davidson%20Testimony.pdf.

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

[4] Google. Greater choice for wireless access point owners. http://googleblog.blogspot.com/2011/11/greater-choice-for-wireless-access.html, Nov 2011.

[5] Google. Android and security. https://developer.android.com/about/versions/android-3.1.html#launchcontrols, Feb 2012.

[6] gpl violation.org. gpl-violations.org project prevails in court case on gpl violation by d-link. http://gpl-violations.org/news/20060922-dlink-judgement_frankfurt.html, Sep 2006.

[7] GPS.gov. Selective availability. http://www.gps.gov/systems/gps/modernization/sa, Feb 2012.

[8] Darren Griffin. How does the global positioning system work ? http://www.pocketgpsworld.com/howgpsworks.php, Jun 2011.

[9] US Coast Guard. Navigation center's navstar gps user equipment introduction. http://www.navcen.uscg.gov/pubs/gps/gpsuser/gpsuser.pdf, Sep 1996.

[10] Robert Lee Hotz. The really smart phone. http://online.wsj.com/article/SB10001424052748704547604576263261679848814.html, April 2012.

[11] Google Inc. Android 3.1 developper launch controls on stopped applications. https://developer.android.com/about/versions/android-3.1.html#launchcontrols, May 2011.

[12] SkyHook Inc. Skyhook coverage area. http://www.skyhookwireless.com/location-technology/coverage.php.

[13] Lorrie Faith Cranor Norman Sadeh Janice Y. Tsai, Patrick Gage Kelley. Location-sharing technologies: Privacy risks and controls, Feb 2010.

[14] Kaspersky Lab. It threat evolution q2 2012. https://www.securelist.com/en/analysis/204792239/IT_Threat_Evolution_Q2_2012/, Aug 2012.

[15] VIPER Lab. Ucsniff: Voip and ip video security assessment tool. http://ucsniff.sourceforge.net/index.html.

[16] Declan McCullagh. Android data tied to users? some say yes. http://news.cnet.com/8301-31921_3-20056657-281.html, Apr 2011.

[17] Graeme McRae. Intersection of two circles. http://2000clicks.com/mathhelp/GeometryConicSectionCircleIntersection.aspx.

[18] The Commission of European Communities. Commission recommendation on the processing of caller location information in electronic communication networks for the purpose of location-enhanced emergency call services. http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2003:189:0049:0051:EN:PDF, Jul 2003.

[19] Zeit Online. Tell-all telephone. http://www.zeit.de/datenschutz/malte-spitz-data-retention, March 2011.

[20] Ben Pan. Apk file infection on an android system, Jul 2012.

[21] Sami Salkosuo. Coordinate conversions made easy. http://www.ibm.com/developerworks/java/library/j-coordconvert/, Aug 2007.