

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
LOUVAIN SCHOOL OF ENGINEERING

Privacy concerns with everyday technologies

Case study of Android phones and wireless cameras

Supervisor:

Pr Gildas AVOINE (EPL/ICTM)

Readers:

Xavier CARPENT (EPL/ICTM)

Vianney LE CLÉMENT (EPL/ICTM)

Thesis submitted in partial fulfilment
of the requirements for the degree

Master 120 in Computer Science

(option *Networking and Security*)

by **Martin Trigaux**



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



Contents

Contents	2
1 Thesis introduction	5
1.1 The context	5
1.2 Case study choices	5
1.3 Thesis organisation	5
1.4 Licence	5
1.5 Acknowledgements	6
1.6 idées	6
I Android	9
2 Introduction to Android	11
3 Localisation using Android	15
3.1 GPS	15
3.2 Wireless access point	17
3.3 Cell tower	19
3.4 Privacy concerns	19
3.5 Personal researches	20
4 Security of Android	25
4.1 Permissions	25
4.2 Installation of applications	27
4.3 Attack schemes	29
4.4 Malwares	32
5 DroidWatcher	35
5.1 Presentation	35
5.2 Mobile application execution	36
5.3 Cell tower triangulation	38
5.4 Technical challenges	40
II Wireless cameras	43
6 Wireless cameras introduction	45
6.1 Analogical cameras	45
6.2 Digital cameras	46

6.3	Web digital cameras	46
7	Security analysis: case study of D-Link DCS-2130	51
7.1	DCS-2130 features	51
7.2	Firmware source code	52
7.3	Installation procedure	53
7.4	Security against traffic monitoring	54
7.5	Guest account	55
7.6	RTSP	57
7.7	Log file	58
7.8	Web camera discovery	58
7.9	PRNG testing	59
7.10	Burglar scenario	61
7.11	Security advises	63
7.12	D-Link reactions	63
7.13	Future researches	63
III	Appendix	67
A	DroidWatcher guides	69
A.1	Mobile application	69
A.2	FAQ	70
A.3	Installation of the web application	71
	Bibliography	73

Chapter 1

Thesis introduction

1.1 The context

The usage of technology is essential in our everyday life nowadays. It keeps us organised and manage our communications in a way we do not even realise it anymore. This thesis does not focus on computers or websites of the internet but on embedded devices used in everyday life. These devices works as a black-box, if they *just work* is usually enough for the usage we have of it. It is very convenient but this convenience has often a price, one is privacy.

The privacy aspect of an embedded device is usually ignored, considered as *good enough*. The confidence in the manufacturer's importance to protect is also a reccurent problem in the privacy scandals.

1.2 Case study choices

As part of this thesis, it has been decided to focus on two types of devices: smart-phones (running Android) and wireless cameras. This choice is part of the current

1.3 Thesis organisation

The thesis is divided in two distinct parts. The content of each part is independent and the order of reading does not affect the general comprehension.

Part I is about the Android system.

Part II is about wireless cameras.

1.4 Licence

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

The text of the current report is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License. This license allow you to copy, distribute, trans-

mit, adapt the work and to make commercial use of the work under the following conditions:

- You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
- If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.

Details and full licence is available at <https://creativecommons.org/licenses/by-sa/3.0/>.

The media used in the report such as graphs, screenshots of interfaces and snapshots of video cameras have been realised, with the exception of explicit source mention in the related text, by the author this thesis. These medias are also licensed under the Creative Commons Attribution-ShareAlike 3.0 Unported License.

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 LICENSE 3.0 licence.

Details and full licence is available at <https://www.gnu.org/licenses/gpl-3.0.txt>

1.5 Acknowledgements

1.6 idées

- Aujourd'hui technologie partout
- L'impact sur la vie privée est souvent négligé
- A décidé de se concentrer sur deux technologies
- Le smartphone
 - Android, très à la mode auj
 - Nombre d'appareils en pleine croissance
 - Nombre de virus en croissance également
 - Contient d'en plus en plus de données personnelles
 - Qu'en est-il de la localisation d'un appareil ?
- La caméra de surveillance
 - Prévu pour nous protéger (argument souvent mit en avant)
 - Apparition de petites caméras personnelles wifi abordables
 - Est-ce que ces caméras sont vraiment sécurisées
 - Analyse en détail d'un modèle en particulier

- Si les faiblesses trouvées sont parfois propre à ce modèle en particulier, représentatif de l'effort mit en avant pour sécurisé ces appareils
- Document structure
 - State of the art
 - android
 - camera
 - for the future

Part I

Android

Chapter 2

Introduction to Android

Structure

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

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

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

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

Why Android ?

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

Market share

Android is the main operating system powering smartphones with a market share of 56.1% in the first quarter of 2012¹. As the smartphone presence increases, it is today most likely that a mobile device is powered by this operating system now and in a close future. The iOS system from Apple is the second most popular system with a market share of 22.9% at the same period of time.

Openness implications

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

Not specific to Android

Although this thesis looks at the Android case, we believe our findings are relevant to other system as well. Today, every smartphone system has location capabilities similar to the one used in Android and these capabilities could be used in a malicious behaviour without the user realising it. The malicious application propagation is directly depending on the platform², but the risks exist and malicious applications are present on any systems that allows such abusive behaviour.

Why the localisation ?

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

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

¹According Mobile Statistics <http://www.mobilestatistics.com/mobile-statistics/>

²The difference of an application publication on the official distribution medium between Android and iOS is studied in Section 4.3.1

possibilities in a transparent way is the aim behind the DroidWatcher application.

The location aspect is only studied in the capabilities of requesting the location of a device. With the exception of the cache file described in Section 3.2.1, the forensic inspection of an Android device has not been studied and could be included in future researches.

Android lexicon

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

ROM: Android is an operating system targeting mobile devices using a Linux kernel. Literally, ROM stands for *Read Only Memory*, the part of the system where the firmware and system applications are stored but the term ROM is often used to designate a new firmware that will replace the existing one. The ROM installed by the manufacturer is usually called the *stock ROM*. CyanogenMod³ is a well-known alternative Android ROM. Its development is driven by the community⁴ and ported to a large number of devices. The installation of a ROM such as CyanogenMod does not modify deeply the system but usually allows more configuration, correct some bugs or extends the possibilities of the device.

Root: on most devices, the owner of an Android device has only a limited access to the system. It can install new applications but not see or alter the content of the device system file where the application is stored. Gaining a root access on a device is a procedure to access all the capabilities of the system and be able to modify it. The procedure for root access varies from device to device since root access protection is left to manufacturers choice and can require complex procedure⁵. Gaining root access is considered sufficient to void the warranty of a device with many manufacturers. Having a root access is required to install an alternative ROM. To realise the experiments described in Section 3.5, a root access was necessary to be able to monitor every aspect of the system.

³CyanogenMod <http://www.cyanogenmod.com/>

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

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

Chapter 3

Localisation using Android

Introduction

The localisation of the mobile device is a key system service of Android. Many applications use this service, as a feature of the application. It allows, for example, to 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 using of an application in a particular region.

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

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

3.1 GPS

3.1.1 GPS signal

The GPS, for 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 is currently 31 working satellites in orbit [6]. 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 3.1. 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).

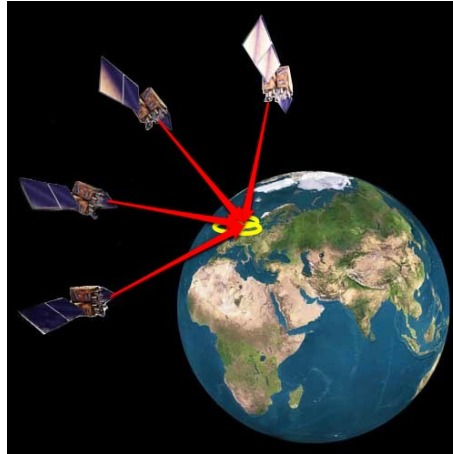


Figure 3.1: Signals from multiple satellites are required to calculate a position
Copyright PocketGPSWorld.com

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

The time to first fix (TTFF) depends on the state of the GPS. In a cold start scenario¹, the GPS needs to retrieve the full almanac. This is done in at least 12.5 minutes [7]. To improve the TTFF, embedded GPS often uses assisted-GPS technology (aGPS) by acquiring almanac and ephemeris data over a fast network connection when available.

3.1.2 Degradation

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

In 2000, Bill Clinton ordered to discontinue the usage of the Selective Availability feature on the GPS satellites, allowing each receiver to perceive the most accurate signal possible. In 2007, the new generation of GPS system called *GPS III* was announced as not capable of producing Selective Availability [5].

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

3.2 Wireless access point

Each wireless access point has a unique identifier². When the wireless option of the phone is turned on, the device can retrieve the surrounding access points. Assuming the geographical coordinates of all the access points are known, it is possible to estimate the location of the user by trilateration.

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

3.2.1 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-driving³ in North America, Western Europe and some Asian countries [10].

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

As they collect data to improve the accuracy of their location services, these companies recently have been subject to criticism. Users wondered about the usage of this database and how it could pose a threat to the privacy of users. The privacy aspect of the localisation section is analysed in Section 3.4.

3.2.2 Methods of data collection

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

- Google Cars
- Crowd-sourcing

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

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

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

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

request a location, data are also transmitted to Google servers. This way, the database of wireless access points and cell towers is always up to date⁴.

3.2.3 Location cache files

Previous cell tower and access point locations are stored in an unencrypted system files. This allows to locate the user quickly and the device is still able to use the network resource, even when not connected to the Internet. Each entry in the cache file is linked with a timestamp representing the date of the retrieval as seen in Figure 3.2.

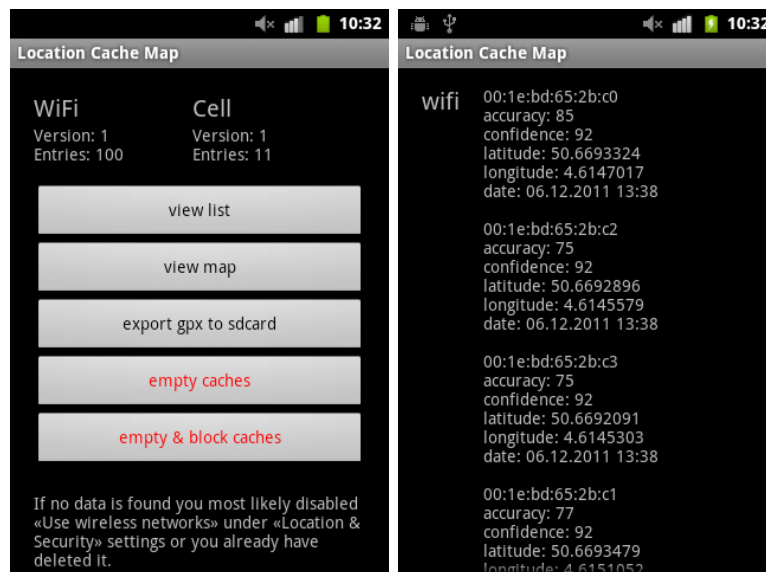


Figure 3.2: Captures from the application Location Cache by Remy Demy

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

To show the ease of retrieving such information, a python script has been developed [2] to parse the content of these two files and produce a GPS trace file in GPX format⁵ representing the approximate movement of the device. However, a root access to the phone is required to access these files⁶. This could prevent

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

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

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

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

3.3 Cell tower

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

The data collection method and cache management system are very similar to the ones analysed in Section 3.2 and will not be discussed here. The location system in Android use usually 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 it allows to locate a device with an approximate accuracy of 1km almost everywhere.

3.4 Privacy concerns

3.4.1 Google Cars

In May 2010, Google admitted to German authorities having collected more than what it was supposed to. In addition to access point's unique identifier, it had "been mistakenly collecting samples of payload data from open networks". These data chunks could include parts of web surf, email, text...⁷. In reaction, the data collected was asked to be deleted and the CNIL (independent French administrative authority) fined Google with €100.000⁸.

3.4.2 `_nomap`

Some users considered the collected data by the Google Cars and Android devices as private. In November 2011, in reaction to criticism, Google created a way to opt out recording of its access point. The proposition of Google is to end the ESSID of the wireless access point with `_nomap`. The next time it is scanned by a Google Cars or an Android device, the access point is removed from the database. Google hopes than over time, the `_nomap` string will be adopted by other location providers [4].

This proposition was received with much of scepticism and did not satisfy the pro-privacy groups. The main complain was the need of an action from the user to explicitly opt out its access point while people wants a way to explicitly opt in

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

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

instead. Many people that are concerned by privacy issues do not have enough technical knowledge to modify the wireless network name. Furthermore, if this string is not universally adopted by other companies such as Apple or SkyHook, conflicting systems can be imagined, preventing a concerned user to fully opt out its access points from commercial databases.

3.4.3 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” [11]. The security researcher Samy Kamkar has also looked at the location requested.

He succeeded to decrypt the request made to Google servers and realised that it contains a unique identifier [8]. The identifier is unique to the cell phone and present in every request. If this string does not directly reveal the identity of the phone owner, it is however possible to tie the string to a specific user and then trace him. He affirms there is no proof the location is anonymised due to the presence of this identifier.

3.5 Personal researches

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

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

3.5.1 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

To facilitate the experiment realisation 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 (battery state...) modified, the database files only are updated. The cache data of Google Maps application has been also deleted. This confirms the assumption concerning the deletion of location data.

3.5.2 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.

3.5.3 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 tried to determine patterns in the requests form to correlate a request with the content of the cache files.

Methodology

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

The network monitoring has been made with the tool *tcpdump*¹⁰ 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 Len=
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=11572
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-f99.1
46 20.290005	192.168.1.12	209.85.148.99	TLSv1	269	Application Data
47 20.293773	192.168.1.12	209.85.148.99	TLSv1	916	Application Data
48 20.347040	209.85.148.99	192.168.1.12	TCP	66	https > 55496 [ACK] Seq=1773 Ack=1320 Win=9536
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=1440
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=1292
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=1724

Figure 3.3: 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 repeating the experiment (results displayed in Table 3.1), the following observations

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

¹⁰TCPDUMP 4.3 <http://www.tcpdump.org/>

were made. Each location requests is composed of two communications with Google's servers. In each communication, a first packet of 271 bytes is sends followed by another 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 3.1: Example of location requests and effect on the cache files

Table 3.1 shows data collected during the experiments realised. **S. cell** represents the number of surrounding valid GSM cell towers at the collect time. 6(1) means there is six surround GSM cell towers detected but only one is valid¹¹. **S. wifi** represents the number of surrounding wireless access points at the collect time. **Req 1** represents the first communication and **Req 2** the second (271 + 644/654 means two packets of 271 and 644 bytes are uploaded and 654 bytes are downloaded from Google's servers). **C. cell** represents the number of cell towers and **C. wifi** the number of wireless access points in the cache files after the location request (numbers between parenthesis represents the number of valid entries in the cache file, as the experiment starts in blank state, the cache files are empty at start).

Figure 3.3 shows an example of collected trace confirming the derived pattern. The packets number 46 and 64 contain 269 bytes and are the initiating packets of a request for cell towers and wireless access points. The packets number 47 and 65 are the surrounding cell towers and access points uploaded to Google servers. The packets number 49 and 67 are the reply from Google's servers containing the coordinates of the known cell towers and access points. These observations do not allow to validate the suppositions concerning the content of the packet but the patterns are coherent to the model explained before.

3.5.4 Experiment 4: Unique identifier

Goal

As mentioned in Section 3.4.3, Samy Kamkar observed that an unique identifier was used in the requests made to Google servers. The presence of this identifier could compromise the privacy of the user as it would allow Google to trace a location requests to a certain user. The purpose of this experiment was to verify this fact.

Methodology

Personal examinations of the cache folder in the system file showed that this string is contained inside a file `gls.platform.key` next to the cache database. When the option *Use wireless networks* in the Android settings is disabled, the content of the cache location folder (containing the wifi and cell cache files as well as this

¹¹Some cell towers are detected but displayed as using a Cell ID and LAC of -1, these are considered as invalid and ignored. These invalid cells have often a very weak signal strength.

identifier) is emptied. When the option is enabled, new cache files and unique identifier are create. The content of the identifier file `gls.platform.key` is different to the previous value every time the option is toggled.

Conclusions

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

Chapter 4

Security of Android

Introduction

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

The aim of this chapter is to explain in detail the current security mechanisms used to protect the users against malicious applications. Using the presented information, a user should be able to reduce his infection risk by adopting simple security principles. The focus of this chapter is the application capabilities and propagation. Different security threads are examined and the 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 to retrieve information from a device without the owner consent have not been analysed here.

These clarifications are essentials to understand the limits and possibilities for the developed *DroidWatcher* application (see Chapter 5) to be effective.

4.1 Permissions

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

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

4.1.1 Technical details

The full list of permissions with a brief description is available on the Android documentation¹. 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 4.1: Example of permission violation log

```
E/AndroidRuntime( 1274): FATAL EXCEPTION: main
E/AndroidRuntime( 1274): java.lang.RuntimeException: Unable
    to start activity ComponentInfo{com.example.gptest/com.
    example.gptest.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 4.1, is shown the Android debugger trace of an application requesting the location of the device using the GPS location provider without having requested the `ACCESS_FINE_LOCATION` permission. If the error is not caught properly, the execution of the application is interrupted and the users receives a notification of the crash of the application.

The permission processed is conceived to control the access to an information and to not a phone characteristic. For example, the `ACCESS_COARSE_LOCATION` permission is not limited to the usage of the high level `LocationManager` methods but is also required for an application to retrieve the surrounding cell towers information (as these towers have a unique identifier, this lower level information could also be used to locate the user²).

4.1.2 Weaknesses

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

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

¹Available at <https://developer.android.com/reference/android/Manifest.permission.html>

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

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

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

4.2 Installation of applications

Unlike iOS where the App Store is the only permitted source of applications³, the Android operating system proposes several ways to install an application.

4.2.1 Play Store

By default⁴, the Android Play Store (formerly named Android Market before its merging with Google Music) is the only source of applications. Once a Google account is associated to the user's phone, it can use the application Google Play Store which lists the available applications and install them quickly. Figure 4.1 shows an example of the interface of the application.

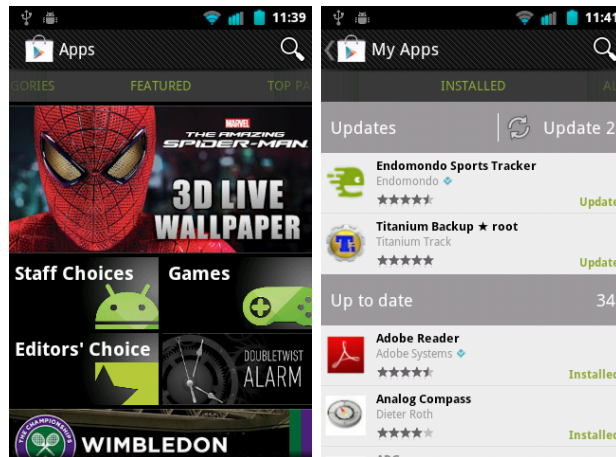


Figure 4.1: Google Play Store interface

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

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

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

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

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

4.2.2 Other sources

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

.apk file

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

A user trying to open such files on his device launches the installation process in the same way as if he was using the Android Play Store. The required permissions are displayed and ask for the approval of the user. Figure 4.2 shows the permission screen when a user tries to install the application DroidWatcher. This is the same screen while using either the Google Play Store or installing an *.apk* file.

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

Alternative marketplaces

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

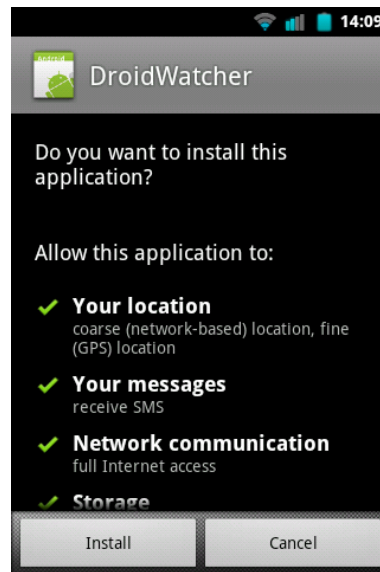


Figure 4.2: Permissions required to install DroidWatcher

Debug mode

If the debug mode of an android device is turned on (done in the configuration settings of the device), interaction between a computer and the device is possible. Using the official Android Debug Bridge toolkit⁵, an application can be installed in a few seconds from a computer without any notifications or user interactions on the device connected to a computer (connection done typically using a USB cable).

4.3 Attack schemes

As the multiple installation procedures make the propagation of application easier, it also allows abuses and permit the propagation of malwares. Figure 4.3 presents the different possibilities for a malicious applications to propagate and be installed on a device. The attack types are described in Section 4.4.1.

4.3.1 Play Store publication policy

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

To distribute an application on the Android Play Store, the registered developer can upload his application on Google Play servers and make it available in a few hours⁶. There is no human control before the publication of an application. The upload of a malicious application would be detected only after its publication which can lead to infected users. In comparison, when submitting an iOS application,

⁵Documentation <https://developer.android.com/tools/help/adb.html>

⁶Official Distribution Control guidelines <https://developer.android.com/distribute/googleplay/about/distribution.html>

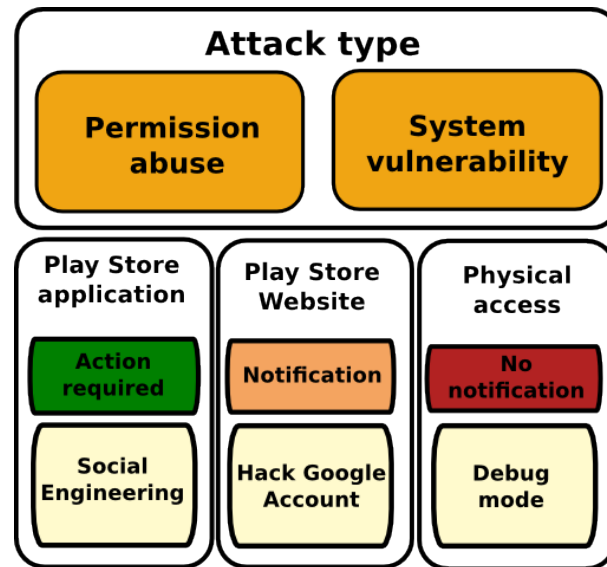


Figure 4.3: Malicious applications propagation possibilities

Apple will review the application which can takes 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 malwares, spywares 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 [3]. 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 recreate the infected apk afterwards [9].

To become a registered developer, the owner of a Google account has only to pay a 25\$ fee and no other information than a name and a phone number. In comparison, to become a registered Apple developer, a minimum of 99\$/year fee⁷ to subscribe to the developer program and more personal information such as credit card information for identity verification are required.

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

As for the Play Store, alternative market places are as secured as the control of their owners over the applications acceptance process. The *Amazon AppStore*⁸ developed by Amazon.com Inc. has an approval process to publish applications similar to Apple regarding iOS applications⁹.

⁷According to Apple Developer Programs <https://developer.apple.com/programs/>

⁸Available in the US only at <http://www.amazon.com/appstore>

⁹Approval Process and Content Guidelines <https://developer.amazon.com/help/faq.html#>

4.3.2 Play Store website

In the case of an attacker gaining access to the Google account of an Android user, it could remotely install any application. This would allow the attacker to do almost any kind of actions the device is capable of. It would be possible to monitor the activity of the user or remotely command the phone. A simple notification is displayed at the application installation that can be unnoticed or not understood as an infection sign by inexperienced users.

4.3.3 Social engineering

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

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

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

4.3.4 Physical access

If a malicious user has a physical access to a device, it can install an application from a computer using the *adb* utility in debug mode. The micro-USB connectivity is an European standard recommendation for smartphone connectivity. It is then easy to connect any smartphone to a computer with this connectivity. If the debug mode is enabled¹⁰, 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.

Approval

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

4.4 Malwares

4.4.1 General malware types

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

The first kind of malware takes advantage of the lack of suspicions from the users and simply ask for permissions allowing a malicious behaviour. This is usually the case for applications sending text messages to overtaxed number or stealing contact information from the address book. This kind of malware tends to be timeless and works as long as the users do not inspect attentively the permission screen whatever the operating system version he is running. As some “honest” applications have a large range of features (that the user may never use), it is common to see such applications asking for many permissions (for example, the official Facebook application requires 19 different permissions¹¹). The reasons of the asked permissions is usually not mentioned by the application makers¹².

The usage of security flaws is possible due to the slow update process. The manufacturers tends to provide only a limited number of version updates if any¹³. A device older than a year is usually not maintained anymore. The only solution for users owning such devices is to install alternative ROMs such as CyanogenMod that provides a longer support for a large range of devices. When a security flaw is discovered and a patch published, only a very small percentage of users benefits from the patch through an update in the months following the discovery. Malware writers can then wrote programs taking advantage of that flaw.

4.4.2 The DroidDream malware

In spring 2011, a malware named *DroidDream* has widely spread across the Android devices. The particularity of this malware was that he used the official Android Play Store (called Android Market at that time). Referring to Figure 4.3, it is classified as using social engineering to exploit a system vulnerability.

The attackers created several developers accounts and malicious applications (above 50 different applications were detected) on the Play Store. The applications used social engineering by taking the name of popular applications and using modified versions of the application to trick the users into downloading them. The malware used exploits effective until the version 2.2 (99% of the devices at that time¹⁴) to break the sandboxing mechanism, root the device and install other applications preventing the removal. The malware has been called DroidDream as

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

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

¹³Computerworld 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>

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

it was set up to run between 11pm and 8am to contact the master server. Due to its ability to install other applications, the Kaspersky Lab's analysts suppose it could have been monetised in the future to be used as a botnet (spam sending, distributed denial of services...).

In reaction to the discovery of this malware, Google activated the *kill switch* which deleted the malware from the user devices remotely. It was the first known example of widely spread command-and-control malware on mobile devices. Researchers estimated the number of infected users between 50,000 and 200,000 devices¹⁵. Variants of this malware called DroidDream Lite have been detected a few months later.

4.4.3 Protection

Observing the large increase of malware applications on the Android platform¹⁶, users and developers wonder about the need of antivirus software. As the antivirus for desktop computers works, these antivirus usually work using a malware database basis. Such softwares would be efficient on antivirus using flaws and derived in several applications such as the DroidDream malware did. However, on the second kind of malicious applications, the efficiency of the antivirus is mitigated as it is very easy and quick to develop applications abusing from the granted privileges.

The PDroid application¹⁷ takes another approach than the antivirus softwares. Instead of detecting the known malicious applications, it allows the users to redefine the granted permissions and revoking them at wish. Another possibility instead of disallowing the access to certain information is to define a fixed or random value (eg: in the case of geographical coordinates). For each application, a notification can be launched at the time the resource granted by a permission is used. This feature can be useful to detect abuse of permissions. However, as the PDroid application works as a intermediate layer between the operating system and the other applications, the application need the root privileges and the user has to apply a patch on the ROM files. These requirements are most of the time not possible on manufactured phones with closed sources ROM and are reserved to users with good computer knowledge. Although it is not applicable to most Android users, PDroid is a possibility of big improvement on the permission model and we can hope a similar model to be adopted in future versions of Android. Figure 4.4 is an example of usage of the PDroid application capabilities on the Facebook application.

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

¹⁶Between 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>

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

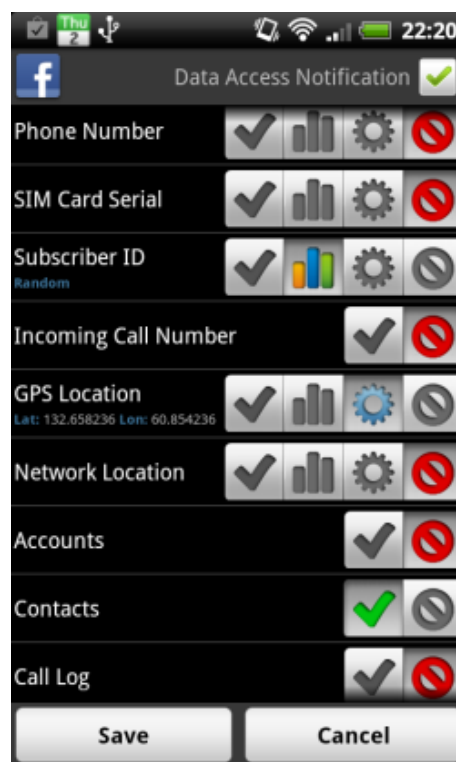


Figure 4.4: PDroid on the Facebook application

Chapter 5

DroidWatcher

5.1 Presentation

DroidWatcher is an application developed in the context of the current thesis. It aims at demonstrating the concrete accessibility of localisation services. By exploiting the official capabilities¹ of the Android operating system, a system monitoring actively the device movement has been developed. In the context of this thesis the application takes only acceptable actions. But of course more invasive actions could be taken by malicious developers.

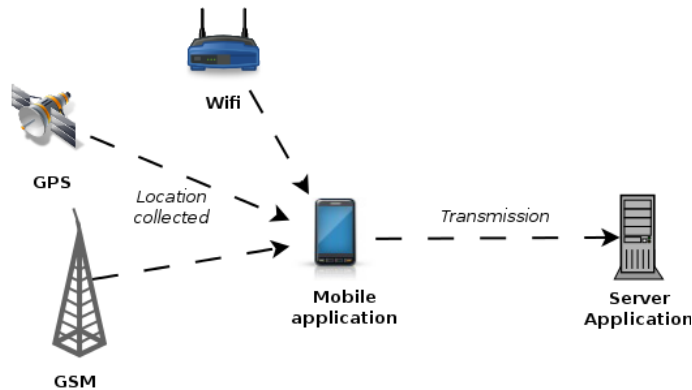


Figure 5.1: DroidWatcher system architecture

DroidWatcher is made of two major components as represented in Figure 5.1:

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

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

¹A known bug has however been exploited for the remote GPS activation feature as explained in Section 5.2.5

5.2 Mobile application execution

5.2.1 Application structure

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

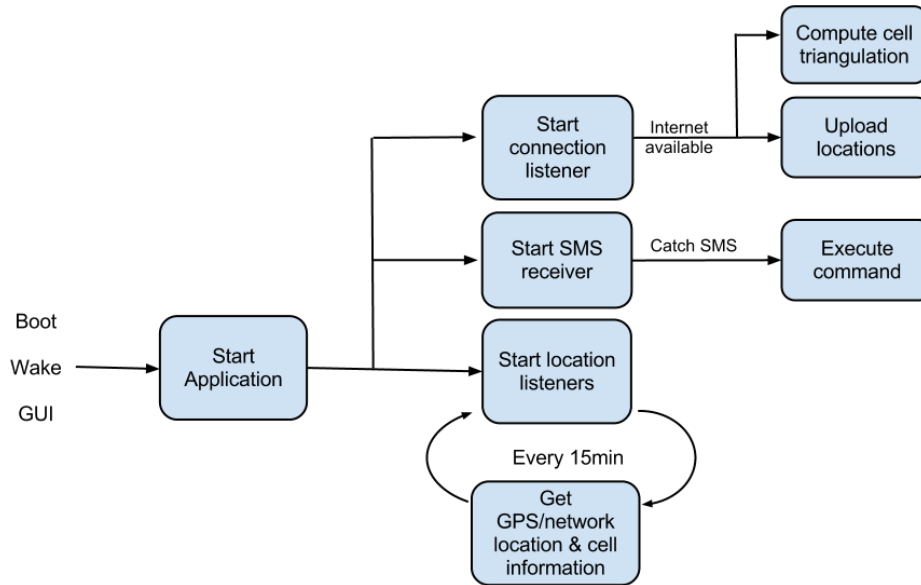


Figure 5.2: DroidWatcher execution process

Figure 5.2 shows the execution of the mobile application. The application starts at phone boot, when unlocked or when the interface is used². When starting, the application has three main components:

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

5.2.2 Location recording

At the installation, the mobile application requests the localisation permissions³. Using these permissions, every specified time interval (15min by default), Droid-Watcher records the location of the device. The application works in background and keeps recording locations when the user does not use its smartphone. Only the most accurate location is kept in an interval of 15min. Both GPS and network⁴ resources are monitored using the official provided methods in Android. If a new

²Except with Android 4.0 or above where the interface should be launched at least once, see Section 5.4.2 for technical information

³See Section 4.1 for details about the permission model

⁴The network resource is defined as the usage of wireless and cell tower access points as described in Section 3.2 and 3.3

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

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

5.2.3 Internet actions

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

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

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

Using the geographical coordinates of the cell towers, the application is able to compute by triangulation of the previous location of the device. The method developed to triangulate a device is explained in Section 5.3.

5.2.4 SMS management

The mobile application provides a SMS-based command interface. These commands allow to configure the application (frequency of retrieval, server url...) and take direct actions related to the location (enable GPS, returns current location...). The application intercepts the received SMS before the main SMS application⁵. 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.

5.2.5 GPS activation

The GPS of a device can be remotely activated using a SMS command. This feature is possible due to a bug discovered in the power control widget⁶. Even if the security flaw has been revealed in April 2010 and a patch released in April 2011,

⁵The 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.

⁶Issue 7890 <https://code.google.com/p/android/issues/detail?id=7890>

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

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

5.3 Cell tower triangulation

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

5.3.1 Scenarios

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

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

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

It has been decided not to differentiate the eventuality of more than three different cell towers as it will greatly increase the complexity of the algorithm without increasing the accuracy of the method proportionally, as intended by the limitations explained in Section 5.4.3.

5.3.2 Determine the position

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

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

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

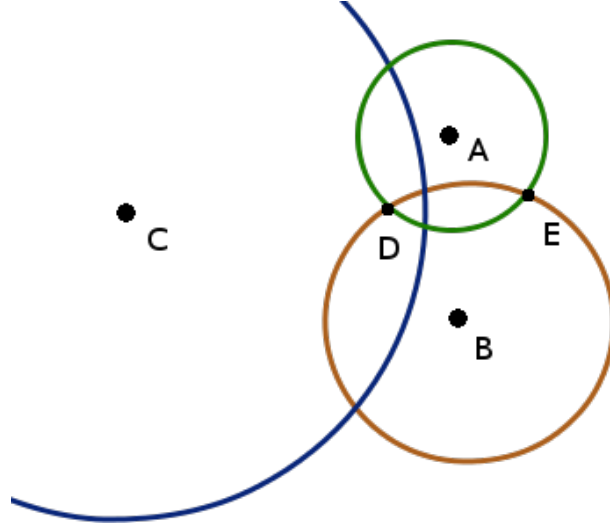


Figure 5.3: Triangulation scenario with 3 GSM towers

The third scenario is illustrated in Figure 5.3. The tower A and B are considered as the closest to the device as received with the highest signal strength, the tower C is the third closest to the phone. For each tower, a circle of a radius size related to the signal strength is drawn. Two points D and E are computed as the intersection between the circles of centre A and B. The point D is determined as the best centre point as it is the one closest to the tower C and is then evaluated as the location of the device.

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

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

5.3.3 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 be. The strongest the signal is, the closest is the device to the tower.

The signal strength collected by the Android device is expressed in *ASU*, for Arbitrary Strength Unit, which is a value between 0 and 31 derived from the signal strength usually computed in decibel⁷. 0 ASU representing the lowest signal and 31 the strongest. The way to express the conversion is made using the following formula :

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

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

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

5.4 Technical challenges

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

5.4.1 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 their running process. The effectiveness of the localisation process done by DroidWatcher is affected by this idle purpose.

This is the case, for example, of the GPS that needs to constantly update of the position. The GPS will sometime stop monitoring the position of the user if it can not get a fix⁸ on the location of the user. This effect is independent of the application but the direct consequence of the system behaviour for battery saving. This issue is often a complaint related to the tracking application (eg: sport monitoring application). A solution is avoid the phone to going to idle state by keeping it in awake state. However, this solution is not used in DroidWatcher as it would have greatly compromised the battery usage of the phone and consequently the effectiveness of the application in monitoring the location the longest and most discrete way as possible.

5.4.2 Android 4.0

As the author of this thesis owns only a device with the Android version 2.3. At the time of development, end 2011, the fourth version⁹ of Android was just released and very few devices were capable of running it. Consequently the testing has been done mainly on devices running the second version of the operating system.

An unexpected change introduced in the 4.0 version¹⁰ of Android is the way a device manage the start of an application in background. DroidWatcher has been conceived to be started when a device is booting or waked from idle. This feature participated in the aim to be fully discrete and that the application was not noticeable without analysis. With Android 4.0, an application can no longer start during the boot or after having been woken up if the interface has not been launched a first time [**boot-restrictions**].

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

⁸See Section 3.1 for the information needed to get a GPS fix

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

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

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

5.4.3 Cell tower triangulation

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

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

Part II

Wireless cameras

Chapter 6

Wireless cameras introduction

Personal wireless surveillance cameras are affordable nowadays. From hotel receptions to home surveillance or supervision of the babysitter¹, many reason 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.

6.1 Analogical cameras

Analogical 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 personal usages. The video camera records analogical pictures and transmits it 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 as 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 there is technically no protection on 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 a few metres.

The usage of this type of camera should only be reserved for non sensible content and be considered as watchable by anybody.

¹Miniature wireless cameras called *nanny cam* are sold on specialised websites with the purpose to ensure the child safety.

6.2 Digital cameras

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

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

The choice of the security mechanism is then directly related to the security of the video stream. 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 convenience by allowing users to access in several different ways (admin interface, mail sending...), all these methods are not equal in term of security. This difference is often unknown and confuses the end user in an incorrect sense of security (eg: setting an admin password to the administration interface during the configuration process does not protect the RTSP² stream in the analysed D-Link camera).

6.3 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.

6.3.1 Connectivity

The deployment of a digital camera may be restricted to a home network while on a local network. This is the default behaviour when a camera such the one studied in Chapter 7 is connected on a wireless network. However, it is possible to make the video available on the Internet. If a port forwarding is configured on a NAT, anyone using the public IP address of the network will be able 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.

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

6.3.2 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 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 of these cameras. It is the case for example of `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 quantity of cameras. By refining the search with other keywords such as *airport* or the name of cities, it may make easier the collect and cross-checking information and facilitate abuses.

6.3.3 Detection of non-public cameras

To end up in Google results, all the cameras have been published on a web page that was indexed by Google. However, it is possible to find camera stream of more private areas such as bars, factories or even homes. These cameras have most likely 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 AXIS³ 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

³AXIS <http://www.axis.com/>

keywords, it is possible to find cameras that were not intended to be published.

Another method to detect cameras are 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 cameras accessible via a public IP address. Some cameras have unique patterns in the URL or in the header of the requests, searching for these patterns allow to detect these hidden cameras. A script detecting the model of D-Link cameras studied in Chapter 7 is described in Section 7.8.

6.3.4 Shodan

Shodan⁴ 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. The case where a camera interface is not protected using an access control mechanism, people using Shodan may find it and be able to watch the video stream. The usage of an access control mechanism does not prevent the indexing of a camera by Shodan but helps to avoid unauthorised accesses. Figure 6.1 shows an example of a search done to find D-Link DCS-2130 camera, finding 195 devices.

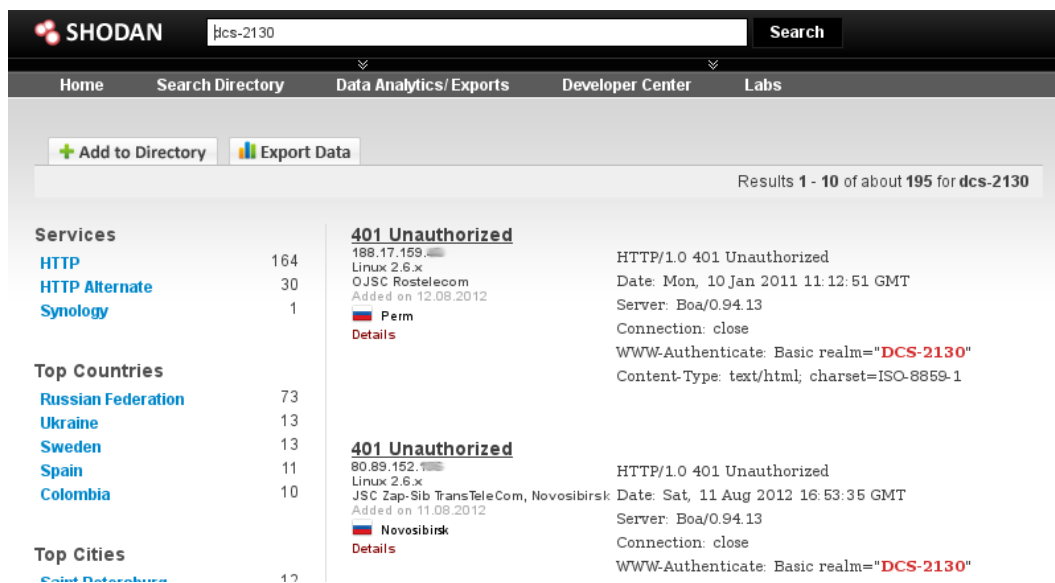


Figure 6.1: Shodan search to find D-Link DCS-2130 camera

6.3.5 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 they are published on the web. This is the case of a set of TRENDnet

⁴SHODAN - Computer Search Engine <http://www.shodanhq.com/>

IP cameras on which a bug has been discovered.

In January 2012, the author of the weblog Console Cowboys, has published some researches he made about 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 `/anony/mjpg.cgi`, any user, even unauthenticated, would receive the video stream from any TRENDnet camera using this firmware as the example in Figure 6.2 shows.



Figure 6.2: Example of video stream accessed on a vulnerable camera

The exploitation of this vulnerability is made easier by the existence of the Shodan search engine mentioned in Section 6.3.4. Using the query `netcam` which is used in the headers of TRENDnet cameras, the search engine lists IP addresses to TRENDnet cameras. Using this lists, 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. However, until new devices with a fixed firmware outnumber the old versions, the proportion of vulnerable cameras may stay large as cameras are very unlikely to be updated by their owners.

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 vulnerable to any curious eyes if they allow the access from the outside of their network. Unlike the results from the Google Search in Section 6.3, 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 7.7.

Chapter 7

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

Introduction

As part of this thesis, it has been decided to analyse the security of one model of wireless digital camera, the D-Link DCS-2130. The camera has been selected as being recent (2011), having good reviews, a large choice of features¹ and being at reasonable prices (around 100€). No previous security researches were found to be made 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 7.1: 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.

7.1 DCS-2130 features

The selected camera model is a D-Link DCS-2130 and is represented in Figure 7.1. The camera supports the following protocols:

¹Technical 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.

7.2 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 [12]. Following that ruling, D-Link published the source code of several of its devices on his website² including the DCS-2130.

Even if D-Link technically provides the source code of the software on his website, many difficulties have been faced to get 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 retrieved³ and the analysis permitted a better understanding of the firmware mechanism as well as the discovery of security issues.

²D-Link GPL Source Code Support <http://tsd.dlink.com.tw/GPL.asp>

³The archive downloaded is available on a public FTP server at <ftp://ftp.dotzero.me/DLink/>

7.3 Installation procedure

7.3.1 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 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 7.5.

7.3.2 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 by anybody listening the traffic. The problem of this protocol is the fact that, in the case of a camera already configured a 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  ..'.4} 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 current username and 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.

7.4 Security against traffic monitoring

As mentioned in Section 7.1, 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 on the same network as the camera and when the wireless network traffic is not encrypted or using WEP⁴. 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.

7.4.1 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 send 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

```

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

7.4.2 URL parameters

To configure the access to external services (email, FTP, DDNS...) 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.

7.4.3 Export configuration

To realise a backup of the configuration of the camera, it is possible to export the 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...).

Behind the fact it is worrying that all these information are unnecessary stored in clear text instead of an encrypted version, this means also that monitoring the traffic will reveals 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.

7.4.4 SSL certificate as a solution

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, either uploaded by the user. Few people will have a valid SSL certificate to upload and it is very likely most of the end users will use the self-signed certificate.

In Figure 7.2, 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 check the validity.

7.5 Guest account

During the analysis of the camera, the exported configuration file mentioned in Section 7.4.3 contains also the information about the different users allowed to access the camera with their access right level as shown below.

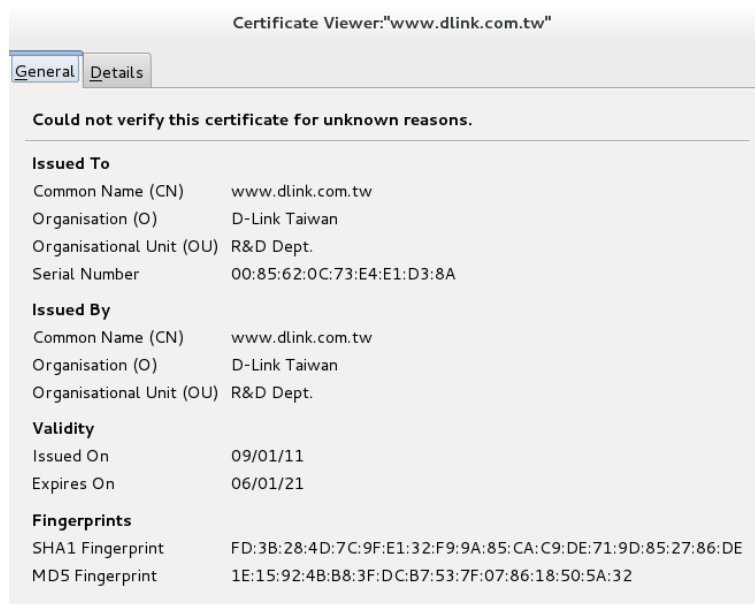


Figure 7.2: Generated self-signed SSL certificate

```
accounts0_name=admin
accounts0_passwd=mypass
accounts0_authority=0
accounts1_name=guest
accounts1_passwd=guest
accounts1_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 his password). While the obligation to set up a password for the administrator account is positive in term of security, the hidden creation of the guest account pushes the end user in an incorrect sense of security.

7.5.1 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 Boa⁵. 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 URI⁶. Using this list, it is possible to identify what the guest account is capable of.

⁵Boa Webserver <http://www.boa.org/>

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

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.

7.5.2 Account suppression

The presence of the guest account is very problematic if a user wants to configure his camera as visible from outside (see Section 7.8). 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 account⁷

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

7.6 RTSP

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. If it is possible to disable the protocol or rename the access path, this protocol does not use any access control mechanism. If an attacker has access to the associated port (554), he can stream the video, even without needing to use the guest account.

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

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

7.7 Log file

As mentioned in Section 7.5.1, 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 of 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 listed URI list. This lack of definition in the request file implies that it will be executed according to the file execution rights, not the one defined in the server list. 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 these information could allow an attacker to retrieve information such as the user habits and presence time.

Like the TRENDnet vulnerability showed in Section 6.3.5, 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.

7.8 Web camera discovery

If an end user configures his router to be able to access his camera from outside of the network (using a port forwarding in the case of a NAT), it opens 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 6.3.3, it is possible to scan a network for the discovery of 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 `dcx_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 (presented in Section 6.3.4) 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 7.3. Through personal experiences, only one camera accessible from the web had the guest account not available but had the RTSP port open. This means there was no DCS-2130 camera available on the web that were found where the video stream was inaccessible. This is the consequence of combination of the unknown presence of the guest account and the facility to detect such devices.

7.9 PRNG testing

As a way to evaluate the security of the camera, the quality of the PRNG (Pseudo-Random Number Generator) has been evaluated. To evaluate the quality of the generated numbers, the tests from the NIST were used⁸. 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 and available in the appendix. This script generates a specified number of connections to the web interface of the camera after having been configured to use SSL. While this script was running, the network was monitored using the tool `tcpdump` recording the communication in the PCAP format. In each

⁸Software available at <http://csrc.nist.gov/groups/ST/toolkit/rng/index.html>.

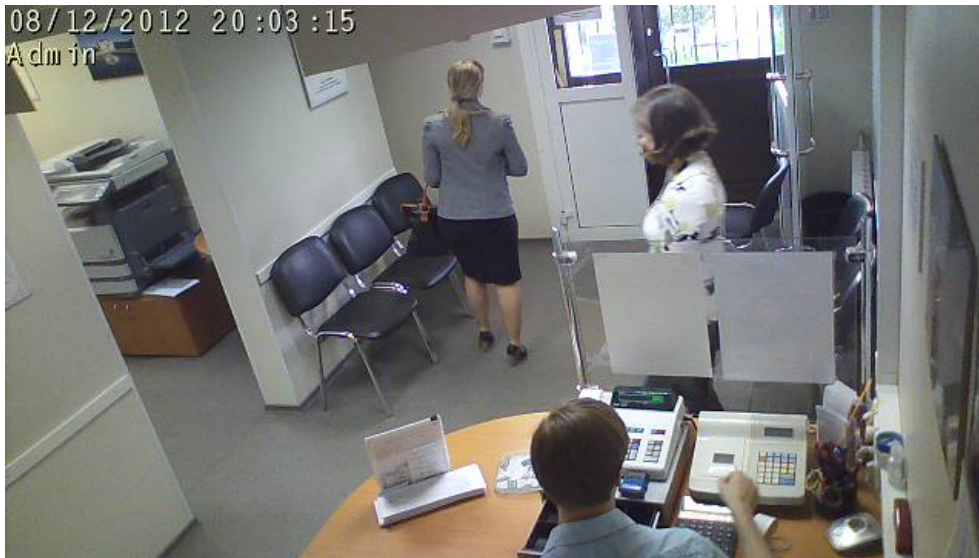


Figure 7.3: Snapshot from a DCS-2130 discovered while scanning an IP range

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 time 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 were that, in every tested run, the entropy of both the laptop and the camera were sufficient to pass the NIST tests. If the tests realised on the laptop were better than the ones using the camera, both were still enough to be considered as random enough. 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.

7.10 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 detailed below depending of the situation of the burglar. Not every case may lead to an abuse of the camera.

7.10.1 Physically close to the camera

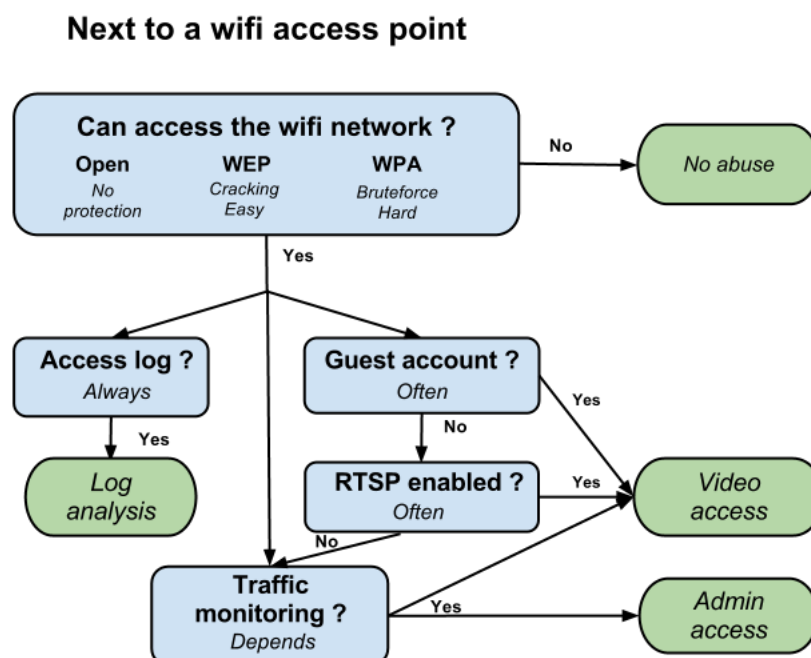


Figure 7.4: Case where the burglar is next to the access point

This first scenario represented in Figure 7.4 is the case where the burglar is in the range of the wireless camera while planning his visit. To be able to abuse from 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 harder⁹

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 occupiers, motion detection system... (always possible)

⁹The wifi penetration techniques are not detailed in this section and left to the reader researches.

2. Access the video stream using RTSP or the guest account to spy the occupiers 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 the camera surveillance mechanisms such as motion detection during his future visit. The access to the video stream helps also the burglar to detect the precise emplacement of the camera and avoid getting caught on tape during his misdeed.

7.10.2 Web access to the camera

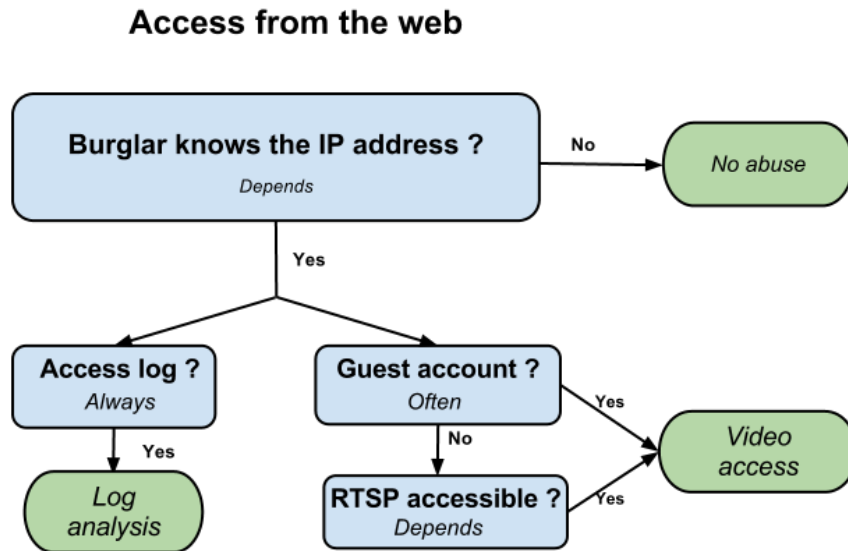


Figure 7.5: Case where the burglar access the camera from the web

The second abuse case represented in Figure 7.5 is when the camera is accessible from the web using the public IP address. The requirement 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 requirement are met, two abuses are now possible:

1. Access the log to retrieve information such as presence hour of occupiers, motion detection system... (always possible)
2. Access the video stream using RTSP or the guest account to spy the occupiers 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 were discovered as fully available from the Internet with the RTSP port open.

7.11 Security advises

To conclude all 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 environment¹⁰.
- Cleanup regularly the cache file.
- Avoid configuring the camera when not necessary, especially using the installation utility.

7.12 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 about these concerns. Both methods have, unfortunately, not been led to an answer.

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

7.13 Future researches

For future researches concerning this camera, there are several points where researches were started but did not reach to a conclusion and could be continued in future work.

¹⁰Wired connection, no potential attacker on the network...

7.13.1 Other models of cameras

The mentioned 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.

7.13.2 Port 1010

While scanning the opened port of the device, every opened port had a clear purpose (HTTP, RTSP, UPnP...) with the exception of the port 1010. This port is opened but during the tests realised this port was never used, neither did it replied 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.

7.13.3 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.

Conclusion

- Android
 - Future ?
 - Ouverture -> risque sécurité
 - Parallele Windows

Part III

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 CDROM¹ 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 panel 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 5.4.2, 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.

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

BIGBRO : starting code for a command.

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

Examples of correct messages:

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

Examples of incorrect messages

- BIGBRO GPSON
- Ping!

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

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

A.2 FAQ

A.2.1 What data is collected by the application ?

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

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

A.2.2 When run the application ?

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

A.2.3 What is stored on the phone ?

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

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

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

A.3 Installation of the web application

To watch the collected information, the DroidWatcher collecting website can be installed on your own web server. The server use the python framework Django 1.3². 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 scope 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 -xzf django.tar.gz
$ cd Django-1.3.1
$ sudo python setup.py install
```
3. Extract and deploy the DroidWatcher Django application from the Droid-Watcher package

```
$ tar -xzf 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.

²Available at <https://www.djangoproject.com/>

7. Generate the database. In the application folder, execute
\$ `python manage.py syncdb`
and choose an admin password.
8. Restart the apache module
\$ `sudo service apache2 restart`
9. Access the received location by going to `http://SERVERURL/admin` to log in
and then access to the recorded location at `http://SERVERURL/`

Bibliography

- [1] Anonymous. *Trendnet Cameras - I always feel like somebody's watching me*. 2012. URL: <http://console-cowboys.blogspot.be/2012/01/trendnet-cameras-i-always-feel-like.html>.
- [2] Magnus Eriksson. *Android location dump*. URL: <https://github.com/packetlss/android-locdump>.
- [3] Google. *Android and Security*. 2012. URL: <https://developer.android.com/about/versions/android-3.1.html#launchcontrols>.
- [4] Google. *Greater choice for wireless access point owners*. 2011. URL: <http://googleblog.blogspot.com/2011/11/greater-choice-for-wireless-access.html>.
- [5] GPS.gov. *Selective Availability*. 2012. URL: <http://www.gps.gov/systems/gps/modernization/sa>.
- [6] Darren Griffin. *How does the Global Positioning System work ?* 2011. URL: <http://www.pocketgpsworld.com/howgpsworks.php>.
- [7] US Coast Guard. *Navigation Center's NAVSTAR GPS User Equipment Introduction*. 1996. URL: <http://www.navcen.uscg.gov/pubs/gps/gpsuser/gpsuser.pdf>.
- [8] Declan McCullagh. *Android data tied to users? Some say yes*. 2011. URL: http://news.cnet.com/8301-31921_3-20056657-281.html.
- [9] Ben Pan. *APK File Infection on an Android System*. 2012.
- [10] *SkyHook Coverage Area*. URL: <http://www.skyhookwireless.com/location-technology/coverage.php>.
- [11] *Testimony of Alan Davidson, director of public policy at Google*. URL: <https://docs.google.com/viewer?a=v&pid=explorer&chrome=true&srcid=0BwxyRPFduTN2NmI2NGVjMWUtZDgONCOONGI5LWJlYTctNmI4MGQ2YmIzYzUz>.
- [12] gpl violation.org. *gpl-violations.org project prevails in court case on GPL violation by D-Link*. 2006. URL: <http://gpl-violations.org/news/20060922-dlink-judgement-frankfurt.html>.