Universidade de Aveiro Departamento de
2014 Eletrónica, Telecomunicações e Informática

Daniel Borges

Privacidade no Sistema Android

Universidade de Aveiro Departamento de
2014 Eletrónica, Telecomunicações e Informática

Daniel Borges

Privacidade no Sistema Android

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requesitos necessários à obtenção do grau de Mestre em Engenharia Eletrónica e de Telecomunicações, realizada sob a orientação científica do Professor Doutor André Ventura da Cruz Marnoto Zúquete, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro e do Doutor João Paulo Barraca, Assistente Convidado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president	**Tomás António Mendes Oliveira e Silva** Professor Associado do Departamento de Eletrónica, Telecomunicações e Informática, Universidade de Aveiro (por delegação do Diretor de Curso)
vogais / examiners committee	**Carlos Nuno da Cruz Ribeiro** Professor Auxiliar do Instituto Superior Técnico, Universidade Técnica de Lisboa
	João Paulo Silva Barraca Assistente Convidado do Departamento de Eletrónica, Telecomunicações e Informática, Universidade de Aveiro (coorientador)

agradecimentos / acknowledgements

Em primeiro lugar, agradeço aos orientadores Prof. Dr. André Zúquete e Dr. João P. Barraca pela oportunidade de realizar este projeto, pela total disponibilidade, constante motivação e conhecimento transmitido, pela criatividade e apoio prestado durante execução desta dissertação.

À Universidade de Aveiro, por albergar os recursos materias e humanos fundamentais para a minha aprendizagem e evolução.

Aos meus amigos, que sempre estiveram presentes nos melhores momentos e também naqueles mais difíceis, não só ao longo deste trabalho mas principamente nestes últimos anos.

À minha família e namorada, por estarem sempre presentes e construírem o suporte necessário para ser feliz todos os dias.

Por último, mas com certeza o mais importante, agradeço aos meus pais pelo incentivo, apoio incansável e principalmente por se preocuparem tanto com o bem estar dos filhos. Foi o seu esforço e dedicação que me permitiram ultrapassar esta etapa e me fizeram crescer ao longo da vida.

Obrigado!

Resumo

O desenvolvimento computacional tem, nos últimos anos, conduzido a uma massificação da utilização de dispositivos móveis, dispositivos estes muito evoluidos não só em poder computacional como também na capacidade de consultar e armazenar todo o tipo de dados e, com isto, a possibilidade de ataque ao sistema e à privacidade do utilizador aumenta. Entre os sistemas operativos mais usados nos dispositivos móveis encontra-se o Android, com uma quota de mercado muito significativa. Este sistema aproveita as enormes vantagens da utilização de código aberto, contando assim com uma evolução tremenda, no entanto, como em todos os sistemas operativos, torna-se necessário adaptar e atualizar o sistema para corresponder às exigências do mercado. No que diz respeito à segurança, o Android tem algumas falhas, e sendo que por um lado, o número de utilizadores comuns não pára de crescer, por outro lado, existe a necessidade de adaptar a segurança do sistema ao mercado empresarial bem como, instituições e organismos governamentais que possuem padrões de segurança elevados. Nesta linha de pensamento, e fazendo uma análise ao sistema operativo, é obrigatório tornar o Android mais seguro construindo mecanismos que permitam zelar pela privacidade do utilizador.

Este sistema operativo permite que se instalem aplicações das mais variadas fontes e isto, aliado a uma disponibilização gratuita das mesmas, acarreta muitas vezes custos à privacidade do utilizador já que estas aplicações acedem a recursos privados de que não necessitam.

Pretende-se então com este trabalho estudar mecanismos de confinamento e ilusão que ofereçam um controlo individual e eficiente sobre que aplicações acedem a determinado conteúdo ou recurso físico. Criando para esse efeito, um perfil falso e distinto do perfil do utilizador, mas coerente e o mais realista possível, passando apenas as informações desejadas às aplicações que tentam aceder a informações privadas sem que delas necessitem para o seu normal funcionamento.

Abstract

Computational development has, in recent years, led to a massive spread of mobile devices, these devices are highly evolved not only computationally as also in the ability to access and store all types of data, thus, the possibility of system or privacy attacks increases. Android is among the most used operating systems on mobile devices, with a very significant market share. This system takes advantage of using open source software, relying with a tremendous evolution, however, as in all operating systems, it is necessary to adapt and upgrade it to meet the market demands. With regard to security, Android has some flaws and this added to the number of ordinary users constantly increasing or the need to adapt the system security to a enterprise level, is required to make Android more secure, building mechanisms to ensure user's privacy.

This operating system allows to install applications from a variety of sources, this, and free applications, often implies costs to the user's privacy since these applications access the private resources that do not require.

The aim of this work is to study mechanisms of confinement and illusion offering individual and efficient control over which applications access certain content or physical resource. For this purpose, we create a false profile but coherent and as realistic as possible, granting or denying access to the data we want to applications that attempt to access private information without the need for their normal functioning.

Contents

Contents **i**

List of Figures **iii**

List of Tables **v**

1 Introduction **1**

1.1 Context 1
1.2 Issues and motivations 1
1.3 Aim of the Dissertation 2
1.4 Outline of the dissertation 2

2 Context **5**

2.1 Android 5
2.1.1 History 6
2.1.2 Architecture 6
2.1.3 Security 11

3 State of the art **13**

3.1 Related work 13
3.1.1 Security Controls for Android 13
3.1.2 Security issues on Android 14
3.1.3 Solutions presented 15
3.2 Conclusions 19

4 Proposed Solutions **21**

4.1 Privacy 21
4.2 Mobile Systems 21
4.3 Deception 22
4.4 Solutions 22
4.4.1 Early Approach 23
4.4.2 Final Approach 25

5 Implementation **29**
5.1 Xposed framework 29
5.2 Contacts 30
5.3 Identification 34
5.4 Location 36

6 Future Work **39**
6.1 Extension of the work 39
6.2 Data Network 39

7 Conclusions **41**

Bibliography **43**

List of Figures

2.1 Android system architecture [2]. 7
2.2 Android activity life cycle architecture [1]. 9
2.3 Android content provider architecture. 10

3.1 Component-based Permission Attack. (Adapted from [10]) 15
3.2 TISSA block diagram [22]. 16
3.3 Xprivacy manages the data provided to the caller application. 18

4.1 Polygon representing country's borders. 26
4.2 Access control for the desired solution. 27

5.1 Contact list retrieved from contacts database. 31
5.2 Contact list separated into arrays. 31
5.3 Arrays mixed with Fisher-Yates algorithm. 32
5.4 New mixed contact list. 33
5.5 Structure of a phone number and the worst case scenario. 33
5.6 The implementation of a new contacts provider on XPrivacy 35
5.7 Example of location feature. 37
5.8 The common path for the two cities. 37

List of Tables

4.1 Most common newborn names in Portugal during 2013 [5] 24
4.2 Most common newborn surnames in Portugal during 2013 [5] 24

Chapter 1

Introduction

1.1 Context

In the world we live nowadays, we carry our social life in our pockets, we use technology to communicate and to be online all day long. Everything is connected and we have dozens of connections on our hands, inside our smart-phone. Losing control over our data can be devastating. Mobile devices are the primordial way of connection to internet and there are thousands of applications available to several mobile devices platforms. Focusing on Android operating system, it is evident the high number of free applications, however this has a cost to the user's privacy. In some cases, private data is compromised since an app can access contacts or user location even if it doesn't need it, to normal functioning.

So, it is important to create a solution which can put that private data inside a box and control who accesses the box content. Controlling which apps accesses which data, can be an interesting approach to this issue.

1.2 Issues and motivations

Android operating system has applications that run over it. Such applications are relatively easy to build, in a matter of days a programmer can learn how to build an Android application, however, this easiness is the same to malicious programmers. Those applications are build by both companies or individual ones and this can easily lead to inclusion of malicious apps on Android devices. These apps can be powered with mechanisms that can leak the entire contact list, the user location or specific digits that can identify the user. This is a major issue to user privacy, since it can reveal user habits by providing user location during entire day and also the possible access to contact list can compromise other users privacy.

There is a clear motivation to achieve a solution to this difficult situation. So, the improvement of mechanisms able to fight privacy invasion, putting private data away from other hands and making safe devices, is a current concern.

1.3 Aim of the Dissertation

This dissertation aims to improve Android tools against privacy data leakage. Being Android operating system present in more than half of mobile devices market share, is important to ensure that Android users have the tools to prevent data loss. Every user has a contact list that sums up dozens or even hundreds of contacts and this information is confidential. The leakage of such data is definitively compromising.

In this dissertation will be explored the way Android treats personal data and will be provided a simple, but efficient, mean to deal with the problem using confinement and illusion. The applications chosen by the user will be provided with fake data, this data must be coherent and will create the illusion of a real profile.

1.4 Outline of the dissertation

This dissertation is divided in seven chapters, the present one is a brief introduction where is shown the context and the aim of the dissertation. The theme of this work is introduced here with context integration. Finally are shown the analyzed issues and the motivation to find a solution for the problem, ending with the concrete objective of this thesis.

Along chapter 2 will be given an overview about Android operating system history and the way it's changing with focus in it's architecture and security system.

Then, in chapter 3, is presented the work related with the thesis. There is the work made in the early stages and some conclusions taken.

Next chapter is dedicated to the proposed solutions. There is a brief introduction to privacy and why this work is important due to private data leakage on mobile devices. In this chapter, are the proposals to solve the issues described above.

In the fifth chapter is the final implementation of the solution found. There are all the steps made to build a working and effective solution.

Chapter 6 contains the future work, this is the possible work to do in this area of knowledge.

The last chapter is the conclusion chapter, here were written some conclusions taken from the execution of this research work.

Chapter 2

Context

The computer industry has evolved drastically in recent years and so the storage and access to private data. If once we had stored on computers lots of personal information such as documents, emails, photos and videos, now in the mobile device's era, we have in every device not only the previous data but also contact data and even the ability to know the actual location of the device. This data is totally private and its leakage can be a serious compromise to the user privacy.

This study relies on the Android operating system since it is the most used among mobile devices.

2.1 Android

The Android operating system became the most popular solution for mobile devices OS's [4]. This platform separates the hardware from software, allowing developers to create rich content available on countless devices. This feature allows a real ecosystem between developers and consumers.

Being an open source platform has magnified Android's power. All the code is open and accessible to anyone interested, any developer has access to each and every part of Android structure. This applies also to manufacturers that can easily port Android to their devices or even include some code over Android base code for adding new features to devices. The user is the most benefited from these openness because is provided with a powerful and beautiful platform created by a huge community.

2.1.1 History

Andy Rubin, Rich Miner, Nick Sears and Chris White founded Android, Inc. on 2003 in Palo Alto, California. Their motivation was to develop an operating system for mobile devices making them more powerful and user friendly. Originally, Android was made to work on digital cameras but soon they realize that there was a bigger market for their product, and they could make a solution capable of compete with the other products in the market such Symbian from NOKIA and Blackberry from RIM.

In 2005, Google decided to invest in mobile devices and acquired Android, Inc.. After that, they started a rail making efforts to form the Open Handset Alliance (OHA) aiming to build a better mobile phone and in 2007 Android was announced. The first smart-phone based on Android was launched on 2008. Since then, Google has been the member of OHA that has contributed the most to develop Android. Google hosts the source code and documentation since the very beginning and launched a Software Development Kit helping developers around the globe to make apps easily even without having a device. Google also organizes annual conferences to developers to talk about new features allowing developers to interact and share knowledge. On the same line, Google has provided some of the most used apps on Android, apps like Gmail, Chrome, Maps or even Youtube were developed by Google.

In recent years, Google created the Nexus series in collaboration with manufacturing partners, they released several hardware devices promoting Android and it's features [8]. Android has become the most used operating system on mobile devices [4]. Google released numerous software updates which improves the operating system by adding new features and fixing bugs from previous versions.

2.1.2 Architecture

This operating system was designed specifically for mobile devices, having in mind their constrains that wouldn't probably change for the foreseeable future. In one hand, the battery of these devices is very limited and for this reason, every step must have in mind battery life. On the other hand, but also related with battery, the CPU of mobile devices isn't powerful as the ones on computers, finally there is the problem of limited memory available. Android operating system was designed to run on low resource devices and every detail must be considered.

Android is divided, essentially, in four layers, as shown on figure 2.1, each one of these

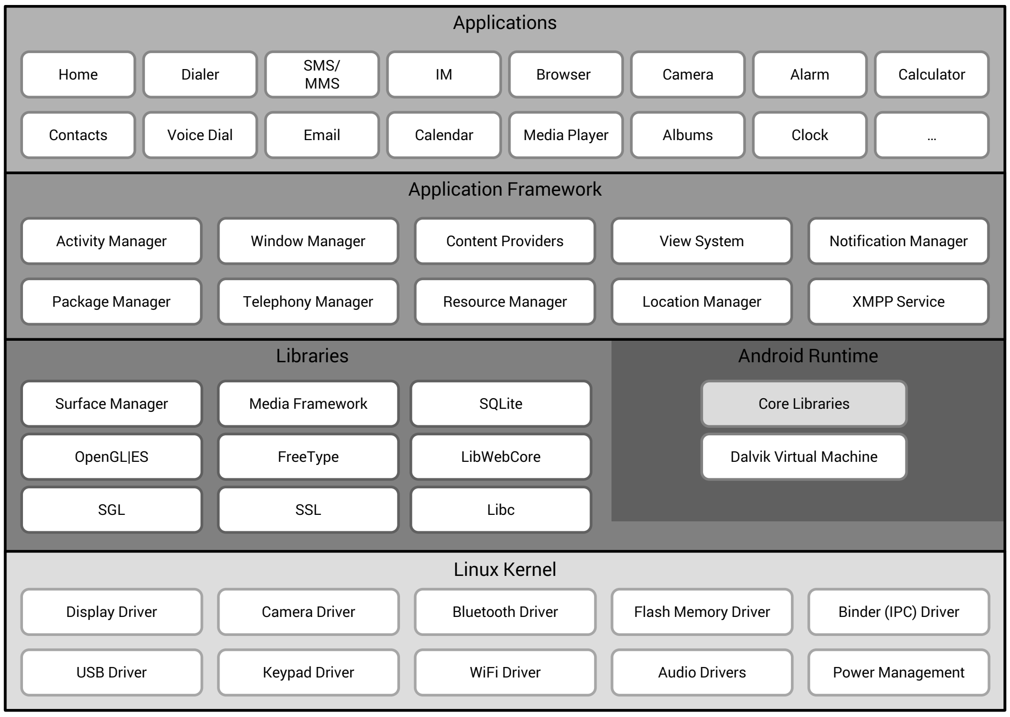

Figure 2.1: Android system architecture [2].

layers is equal important for the efficient execution of the operating system.These layers are on top of each other with lower-level layers providing services to the upper-level ones [19].

Beginning from the bottom we have Linux kernel, this layer is the interconnection between hardware and software providing an effective hardware abstraction. Here are the device drivers which are the tools that make every component of the device work as intended. This layer is similar with the one existing in desktop computers running Linux. It is the base of the operating system.

At the next layer is the runtime layer including native libraries, these act as a translation between the kernel and the layer up to this one, the application framework [15]. Written in C/C++, these libraries provide many services that are available to applications by the different components of the system. Some of these services are commonly used to provide: graphics to device's screen, drawing engines with 2D and 3D graphics, data-store technology of Android platform (SQLite) or even media services as playing music or video.These native libraries run as processes within underlying Linux kernel [19]. The runtime layer also includes a runtime component which consists of the core libraries and Dalvik Virtual Machine (DVM). DVM was written to allow devices with limited resources such as mobile devices execute java applications, it works as an operating system within a host operating system and its main feature is the portability, regardless the hardware this VM will execute the written app. This allows developers to write applications once, and be sure that they will execute in every compatible VM despite their hardware differences. Core libraries are present in this layer and also in each app's runtime, they are Android class libraries, I/O and others.

The upper layer is the application framework fully written in JAVA, and here are found the services immediately above the applications we use. Running at this level are the entities responsible for managing the apps on the device and the app life-cycle itself. Here, is also provided the access to basic elements such as buttons and text boxes.

Finally, we have applications running at the top level, which this is the closest to end user. We can find here all the apps we use, from system apps up to third-party apps. Usually, applications have one or more of these four components: activities, services, broadcast receivers and finally content providers.

Activities, represented in figure 2.2, are the visible part of an application, it's the component that a user sees and interacts with. Typically one app has several activities. Besides that, services don't have any user interface. They act like activities and do what activities do but without a user interface performing background processing. The best example to define service is the use of media services to play music while using a different app. Broadcast receivers are components of Android that implement a publish/subscribe mechanism acting as mailboxes for messages from other applications; the receiver gets activated by the triggering of a specific event for which it has subscribed. Finally, as each application runs in its own sandbox, data owned by one application is accessible by that application only and isolated from the other.

Content providers are interfaces that allow different applications to share the same data as illustrates in figure 2.3.

The Contacts provider allows the access to the contacts database from any application with permissions to do so. On the same line, settings provider is used to change system settings using built in the settings application or another application able to do the same [13]. Each application has to declare the components that it will use and these components are specified in the manifest file.

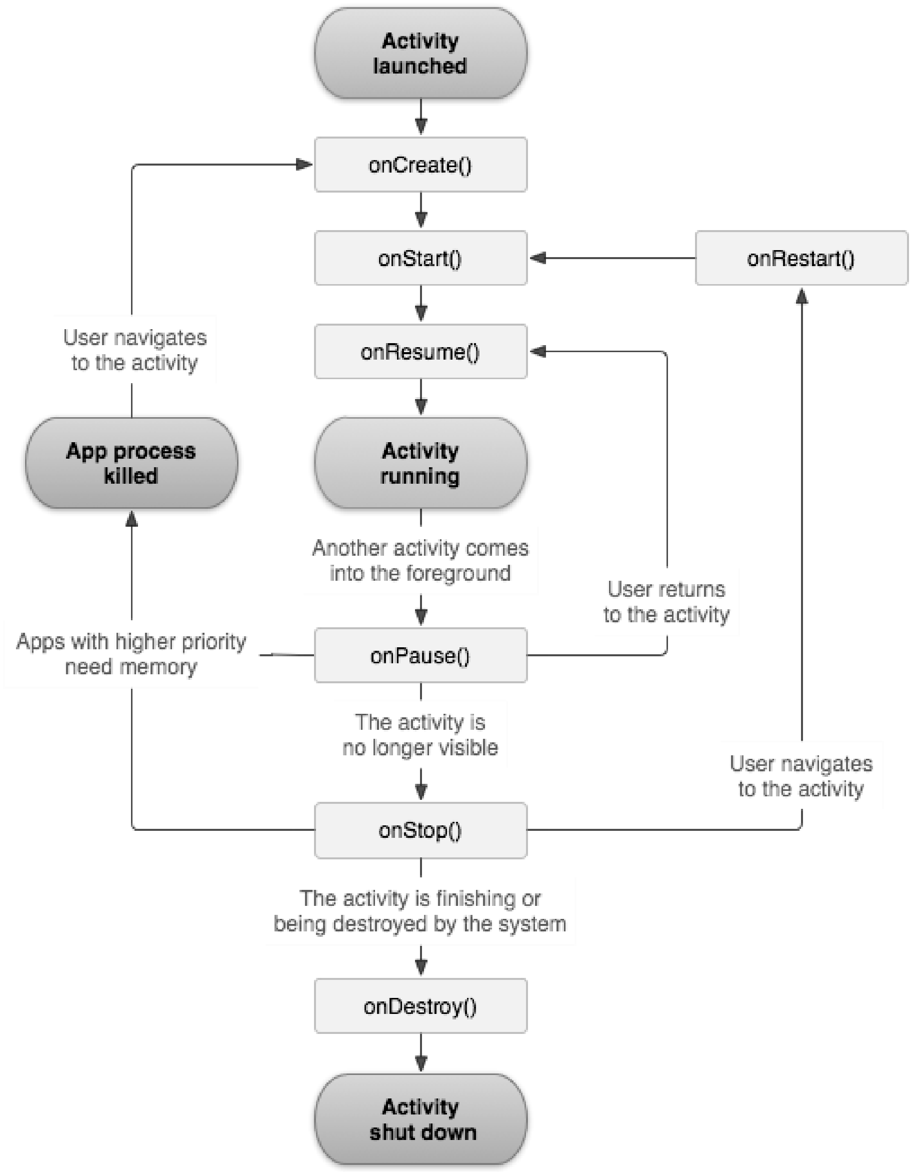
Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Figure 2.2: Android activity life cycle architecture [1].

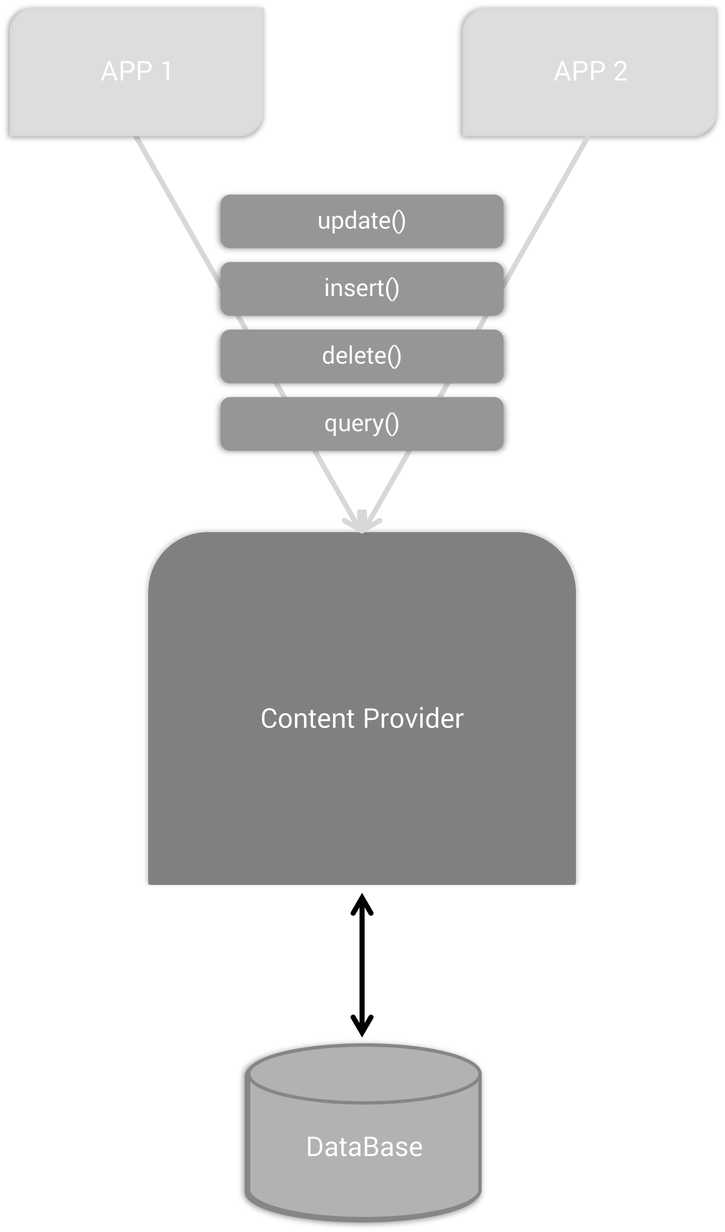

Figure 2.3: Android content provider architecture.

2.1.3 Security

As Android is build on top of a Linux kernel, it shares the same security heart of Linux. All applications run as a separate Linux process with all the characteristics of these processes. Android passes many security concerns to the Linux system. The Linux kernel has been used for years and it's used in sensitive environments [2]. It is a stable and secure kernel trusted by many corporations and security professionals.

The Android platform benefits from Linux stability, specifically from process isolation and a secure mechanism for inter-process communication (IPC) [12]. Each Android package has its own user ID (UID), this creates a sandbox for each application preventing apps from interfering with each other and from accessing files that belong to other processes. All the files on Android follow the Linux permissions mechanism (Read, Write and eXecute). Android provides IPC mechanisms besides the ones present in Linux: binder, that is a cross-process call mechanism, designed for high performance; services, that provide interfaces directly accessible; content providers, referred above and intents. Intent is basically a message that represents an "intention" to do something.

The access control in Android is based on a simple permission label assignment model [18]. This permits to block the access to resources that the application doesn't use or need. The access to the system API is restricted by permissions, applications must request in their manifest to use protected API calls. All permissions have a protection level according with how demanding it is to acquire that permission. There are four distinct levels [6] normal, dangerous, signature and signature or system. Normal level permissions are granted automatically, as they are a lower risk permissions. Dangerous permissions are granted during app installation and if denied, the app isn't installed. Signature permissions are granted if the requesting application is signed by the same developer that defined the permission. Finally the signature or system permissions are granted if application meets the signature requirement or if the app is installed in the system folder.

Recently Google launched the Verify Apps feature, to detect malicious processes running on the device and block third party applications from installing. This feature will run in the background detecting suspicious processes [16].

Some threats to Android security are already exposed and few solutions are presented, this will be explored in chapter 3.

Chapter 3

State of the art

In this chapter we will overview the existing work related with the present thesis. During the early stages of this work a comprehensive search was conducted, and the items found during the research are present in section 3.1. Later, in section 3.2 are presented the conclusions taken after the research phase.

3.1 Related work

3.1.1 Security Controls for Android

Vargas et al. [20] discusses the use of smart-phones in business and its vulnerabilities, it covers Android version 2.3. Android is an operating system well spread among smart-phone users but not particularly in business market and it has some security breaches that should be fixed. In this paper, the authors identify some lacks in Android security model and show some solutions for the identified problems. They classify Android threats in three groups: application, web and network based. It's possible to install applications in many ways: application store, third party stores or even directly by USB connection. This is a huge security lack, as everyone can install applications that can access and share personal data. These applications may have vulnerabilities that allow access to attackers, and they can use unwanted resources or download malware without user's knowledge. The presented solution is to determine the minimum required installed applications on Android in order to run without crash. Another implementation would be the use of Selinux [7], a mandatory access control mechanism preventing applications from access unauthorized resources. Disabling unused services would be another type of defense. At network level, we can find a security issue, the user can be victim of Wi-Fi sniffing or network exploits and the proposed solution is to create a firewall, functionally similar to firewalls present in ordinary computers. The idea of using the same kind of security used on laptops applied to smart-phones, such as encryption, is supported in this article, this would prevent attackers from accessing decrypted data stored in SD card or even in internal memory of the device. In conclusion, the operating system under analysis needs to be fortified in order to be safely used in business environments. The use of firewalls, data encryption, access control and disabling unused services must be implemented in some kind of way to ensure a secure use of smart-phones.

3.1.2 Security issues on Android

In this subsection will be shown several Android security flaws identified in scientific publications.

Gibler et al. [14] studied some potential privacy leaks on Android. They explained the Android architecture and detected a problem. Each application declares the sensitive data and the functionality that it requires in a manifest file, however, it is not clear how sensitive data is used once the application is installed. They presented a framework for automatically find potential information leakage (AndroidLeaks). After the analysis of 24350 applications and as a result of this study, they concluded that 7414 were leaking private information. Android developers must be aware of this result in order to improve their code to provide safer applications.

Yang and Yang [21] proposed an approach based on taint analysis to detect Android information leakage (LeakMiner). This solution allows to detect leakage even before the distribution of applications to users. They analyzed apps from the market and searched for faults that could compromise user's privacy. To achieve the detection was used static taint analysis.

Likewise, Chan et al. [9] used static taint checking and also inter-procedural control flow graph searching to detect malicious access to private data.

Davi et al. [10] exposed permission escalation attacks and explained how an app without permissions can access data that requires permission, as seen on figure 3.1.

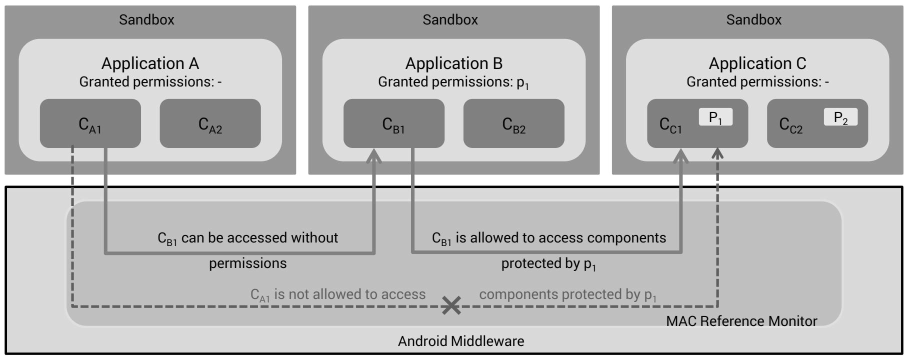

Figure 3.1: Component-based Permission Attack. (Adapted from [10])

3.1.3 Solutions presented

Here are some of the solutions to mitigate attacks against privacy of user data. Do et al. [11] suggests the removal of permissions. The Google itself included a feature called App Ops, however, this is an hidden feature and not available on most recent Android version [3]. TISSA, a tool presented by [22], is the solution we've found that is closer to ours. Here the authors want to provide to the smart-phone user a tool that increases user's privacy. Their approach includes a system that allows to deceive applications by feeding them with garbage. TISSA's architecture is illustrated in figure 3.2.

Xprivacy

Marcel Bokhorst built Xprivacy as a module of Xposed framework. This framework can intercept any call made on Android and then a module may inject its own code, making this framework very powerful. Xprivacy uses the Xposed framework to change a large number of Android functions carefully selected, skipping the execution of the original function or changing the result of the function called.

Xprivacy main objective is to prevent applications from leaking sensitive data by restricting its normal access to data. It separates data by categories, restricting applications

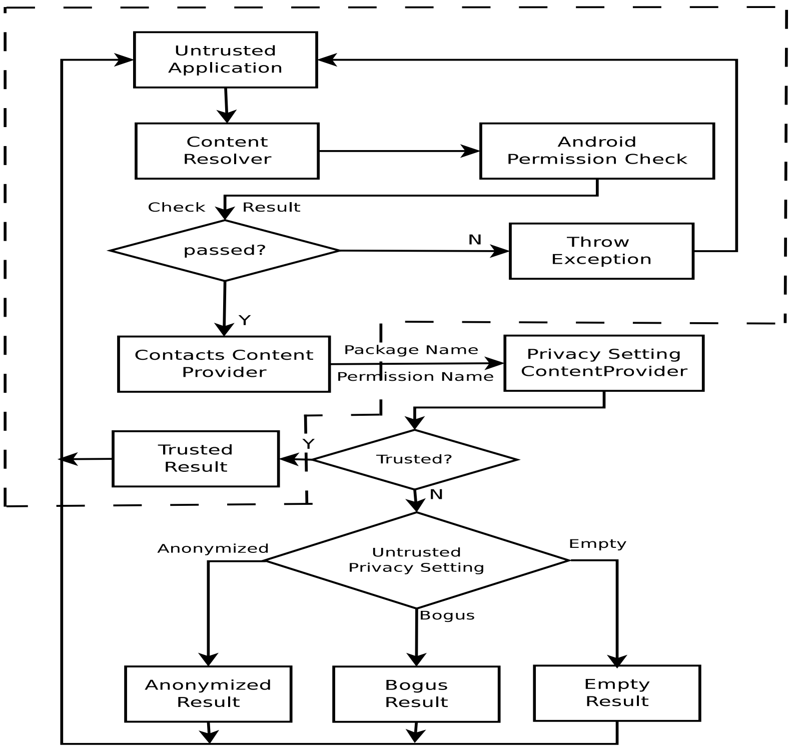

Figure 3.2: TISSA block diagram [22].

from accessing data from a specific category. This module doesn't block any permission given to an application, and for that reason, applications do not crash, they work as before. Xprivacy simply feeds applications with no data or fake data. On figure 3.3 is one diagram with the architecture of XPrivacy. As we see, it uses the hook detected by Xposed framework when a call to query the contacts database is made, based on the defined policy of that application, XPrivacy will feed the application with data from the database or no data at all.

Newly installed applications cannot access any data category by default, this prevents applications from leaking data right after their installation. Xprivacy will ask the user during the application's first use, which categories the app has access to.

XPrivacy identifies possible data leaks, and monitors all applications' attempts to access data and displays all information.

If contacts category is restricted for a determined application, Xprivacy will provide an empty list of contacts. As an example, if People app is restricted on Xprivacy it will show an empty list and display the option to create a new contact.

With regard to identification, it is possible to randomize some numbers inside of Xprivacy, serial number of the device, MAC address, IP address, phone number, country, mobile operator, etc.. But this isn't a smart randomization, it is a simple change of digits by others arbitrarily chosen.

The Xprivacy approach to location data is basically the same taken with identification. It is possible to randomize device's location or even insert one location that will be shown to every restricted applications, this solution suffers from a not so realistic approach.

Xprivacy is very useful for this work, however, it provides applications with fake data or no data and this solution is not realistic. The data provided doesn't cause any illusion, it's clearly aleatory data with no meaning. An attacker can easily find that data is not correct or that it is fake data. As an example, we have the empty contacts list, which isn't normal . In the same way it's not usual a device having the same location for long periods.

In order to complete this study, it was needed to change Xprivacy to feed applications with fake, but at the same time, coherent and realistic data.

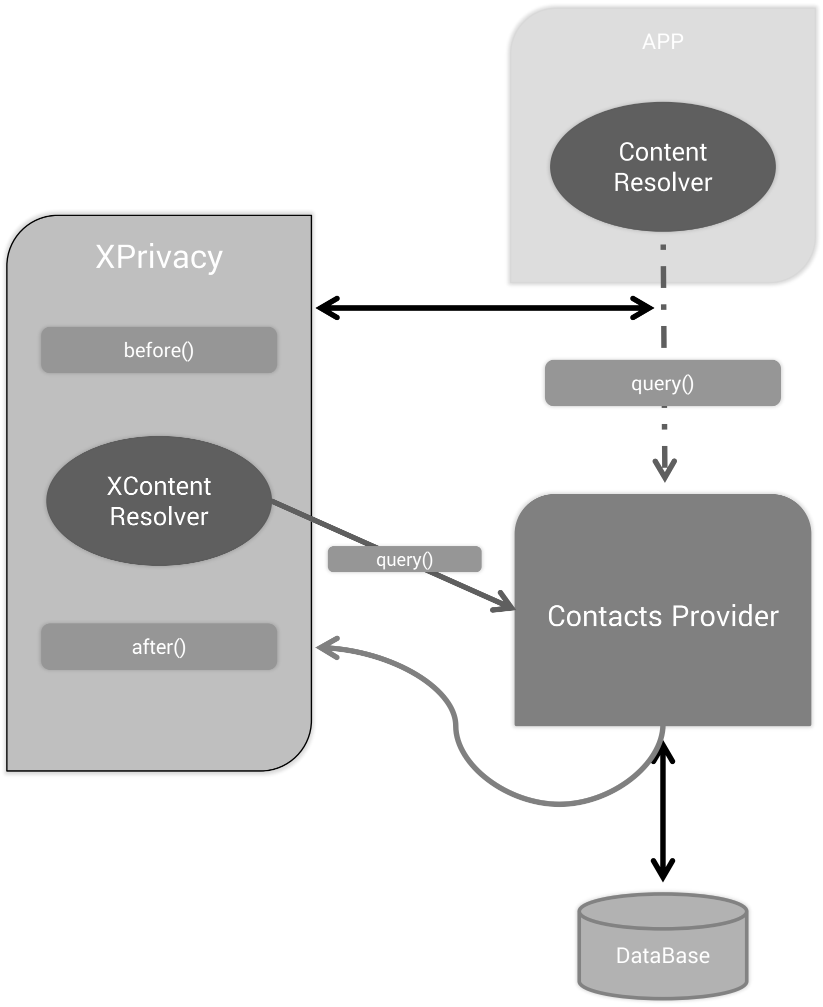

Figure 3.3: Xprivacy manages the data provided to the caller application.

3.2 Conclusions

In this chapter was presented the research work done to this study. Some flaws and critical points of the Android operating system were shown. In the other way, some solutions were discussed to improve the Android security architecture. It's important to note the two works that were found: TISSA and XPrivacy. TISSA uses deception to fool applications by feeding them with garbage. However, it isn't available the way TISSA build the data provided to the applications or the application itself. Xprivacy works similarly to TISSA, it has a policy database to save the rules for each application and it feeds applications with fake data or no data. The fake data that XPrivacy provides is the simple randomization of values. We search for a realistic solution based on confinement, and illusion. We can find confinement in XPrivacy but not the illusion part.

Chapter 4

Proposed Solutions

4.1 Privacy

Nowadays everyone is worried about privacy, more than ever private data access by third parties is a top of the line problem. The focus on this aspect has mobilized people to write about it and to effectively develop some solutions. By definition, privacy is the state of being free from intrusion or disturbance, so, any kind perturbation on someone's life makes an invasion of privacy. Private data can make money roll since there are organizations or even individuals capable of monetize this data, illegally obtained. Since every human being has the right to his own privacy every kind of attack over private life is reprehensible. In the past years, we changed the way we look at privacy at a technological level. The human interconnection has changed drastically, and now, we are a phone call, an email or even a message away from each other. The world became smaller and this brought some concerns, an usual smart-phone user can carry dozens or even hundreds of information about other users and the loss of such data can compromise both, user and his contacts. In other way, this devices are able to obtain their own physical location, this is also considered private data as can reveal user habits. Every smart-phone is provided with a GPS sensor, Wifi sensor and even GSM system, that accurately retrieve user location. If this data is obtained by a malicious person, the user is compromised with such abuse of privacy.

4.2 Mobile Systems

The history of technology show us an abrupt transformation over years. Talking about massive computers that could do some calculations per second is now a remote memory.

The kids that are now born won't probably see a desktop computer like it is today. We took a huge step from desktop computers to laptops but recently, we got really mobile. With absolute remarkable advances in overall technology those laptops are being take aside by smart-phones and tablets. We can assume that these technology wonders will prevail and so, we should support its development as much as we can. There are clearly three major actors in terms of mobile operating systems, those are, Android, iOS and Windows Phone. These OS's are maintained by three giants of tech industry and are present in almost every pocket of citizens from first world countries and for sure in every house.

4.3 Deception

As a subsection of Android Security we have Deception, we can say that this kind of security enforcement is not common but it is an effective way to tackle private data loss. With this feature it is possible to create illusory data that will confuse the attacker and preserve user original data. In this case, illusion is the ability to give fake data to specific components. The data information as a whole has to be precise and coherent, the data has to make sense to the external actor, the attacker. Hence, the attacker is allowed to get into the device and steal information, if this information is coherent and realistic will make the attacker believe he succeeded when, in fact, he has false information.

4.4 Solutions

After the explanation of these concepts it is important to remember the actual aim of this dissertation[1].

This solution basically consists in observe applications' activity and restrict the access to resources that shouldn't be used by them. Having a listener on device's activity, every time an application tries to access any sensible data the system call is intercepted and the app gets a return value different from the real value, normally a empty value or even a fake value, making attacker believe that he has valid data.

As Christopher and Artem identified[17], there are several sources of information on devices, here will be explored three main areas: contacts, identity and location. So, the focus will be on these areas and they will be tackled one at a time.

[1]In this dissertation will be explored the way Android treats personal data and will be provided a simple but efficient mean to deal with the problem using confinement and illusion.

4.4.1 Early Approach

The first approach to solve the faced problem was reached at the beginning of the present study and the aim was to create an illusory profile based on a defined country. Given a list of countries, it was supposed to have a database with information for each country. This information would consist in the most frequent names and surnames in each country, all code numbers existing and the country physical borders. From that information it should be feasible to build a false profile, always being coherent and as much real as possible. For instance, if the country chosen was Portugal the mechanism would push information from the country database and build a contact list based on Portuguese names and surnames, as well as random numbers with the real code numbers and a random location within Portugal borders. This example is explored in the next subsections, but was not the final solution as it needs a potentially large amount of data and was not easily adaptable to other countries.

Contacts

Regarding contacts, the main goal was to build a separate contact list totally different from original one, as applications ask details from contacts list the system would give references the original or to the fake contacts list. The fake list would be made by collecting some of the country best known first and last names and then mixing them together formulating the contact list with names different from the ones existing in the real contacts list. An example is presented on tables 4.1 and 4.2, as we see, the most common names could be used to build a new contacts list. To build this list, we could simply link one aleatory name with a surname and then include a random phone number to create the final contact list entry.

About the phone numbers in the contacts list, they would be composed by acquiring country code and network provider code and then randomizing the last numbers. This solution would provide a contact list away from reality but still coherent with the specified country.

Identity

In terms of identity, it would be based on a previously specified country and then randomizing all possible numbers, as the personal number based on operator call-sign and randomizing all other digits. Regarding other identity values, always based on the chosen

João	António	Maria	Íris
Rodrigo	André	Matilde	Letícia
Martim	Diego	Leonor	Mara
Francisco	Vicente	Mariana	Catarina
Santiago	Manuel	Carolina	Gabriela
Tomás	Henrique	Beatriz	Marta
Guilherme	Leonardo	Ana	Vitória
Afonso	Vasco	Inês	Yara
Miguel	Bernardo	Lara	Camila
Gonçalo	Mateus	Margarida	Ariana
Duarte	Luís	Sofia	Núria
Tiago	Eduardo	Joana	Daniela
Pedro	Alexandre	Francisca	Iara
Gabriel	Leandro	Laura	Ema
Diogo	Rúben	Madalena	Rafaela
Rafael	Filipe	Luana	Benedita
Gustavo	Ricardo	Diana	Bruna
Dinis	Samuel	Mafalda	Filipa
David	Bruno	Rita	Júlia
Lucas	Matias	Sara	Bárbara
Salvador	Nuno	Bianca	Jéssica
Simão	Enzo	Alice	Victória
José	Rui	Eva	Carlota
Daniel	Hugo	Clara	Alícia
Lourenço	Carlos	Constança	Nicole

Table 4.1: Most common newborn names in Portugal during 2013 [5]

Silva	Gomes	Nunes	Reis
Santos	Lopes	Soares	Simões
Ferreira	Marques	Vieira	Antunes
Pereira	Alves	Monteiro	Matos
Oliveira	Almeida	Cardoso	Fonseca
Costa	Ribeiro	Rocha	Machado
Rodrigues	Pinto	Neves	Araújo
Martins	Carvalho	Coelho	Barbosa
Jesus	Teixeira	Cruz	Tavares
Sousa	Moreira	Cunha	Lourenço
Fernandes	Correia	Pires	Castro
Gonçalves	Mendes	Ramos	Figueiredo

Table 4.2: Most common newborn surnames in Portugal during 2013 [5]

country all the other possible digits would be different. All identification numbers have to be identified and then randomized.

Location

Given the referred country borders, a random position within the country would be generated. The borders would be generated at the time country was defined, creating geometric coordinates that approximately could define the country physical borders.

As we see in figure 4.1, a polygon would represent the country's borders with the respective location coordinates defining the limits to choose a user location.

This approach assumes that user could be anywhere inside the country, even in the most inaccessible places, which is not a realistic approach.

4.4.2 Final Approach

The solution described above was unsuitable because it needed big amounts of data for each country, like most frequent native first and last names, call-signs and a big number of coordinates to define each country. The solution we were looking for had to be versatile and adaptive in a way that would be suitable to largest number of users despite their options and living style.

So, the final solution consists basically in creating an illusory profile based on the real data stored in the device. This illusory profile will be shown only to that apps that user may consider potentially malicious or to apps that the user doesn't want or need to share personal data with, figure 4.2.

In the rest of this chapter we will explain the solution that we considered to be the best for creating and providing a fake profile to untrusted or abusive applications

Contacts

In order to create a fake but, at the same time, realistic contacts database, we used as a start point for that database the original contacts list. Having the original contacts, we have a credible source of data and all we have to do is to mix this data in a way that the original contacts are indeterminable. In this case we do not use predetermined names and surnames to formulate the new contact list. Using a simpler but nevertheless effective approach, we build the new contact list by separating original contacts' names and mixing them together again in a way that doesn't allow the original ones to be recognized.

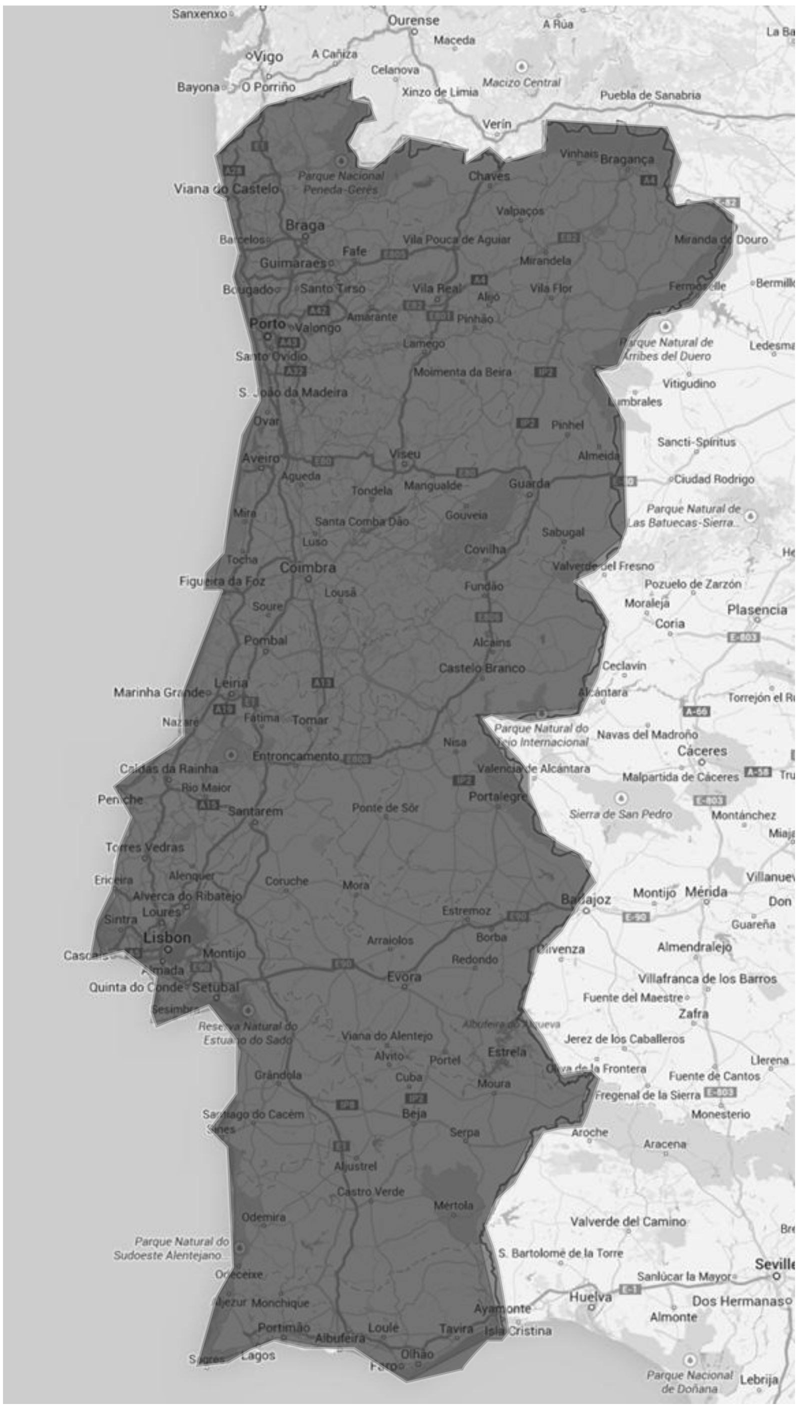

Figure 4.1: Polygon representing country's borders.

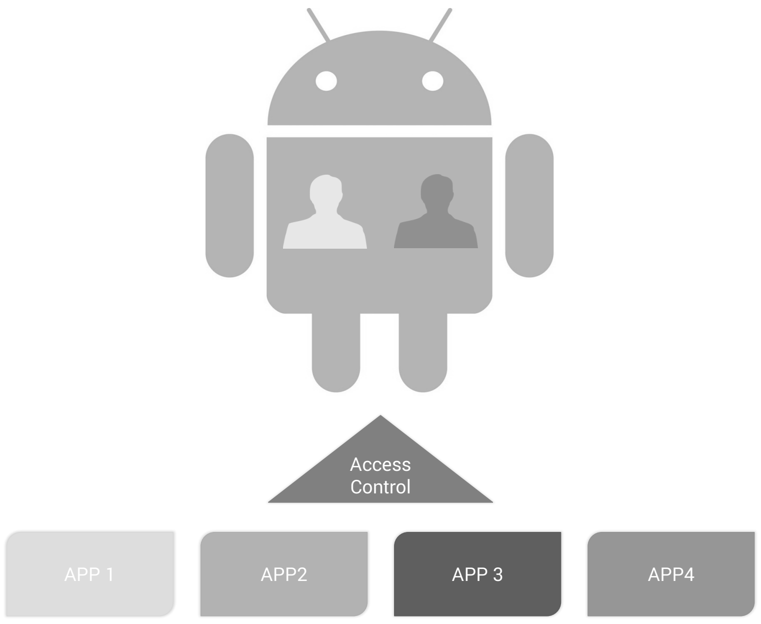

Figure 4.2: Access control for the desired solution.

Regarding contacts' numbers, the option was to keep the first numbers that are usually the country code and the network call-sign and randomize the remaining ones.

Identity

In terms of identity, this solution isn't different from the previous one presented. Having in mind not a specified country but the actual one, given by the user actual location or by the country code existing on IMEI or even using mobile network info available on device, all possible digits would be randomized. In order to be coherent, this solution has to consider all the specifications of each identifying number present on the device. So first, all the identifying numbers that undermine user privacy have to be exposed, analyzed and then randomized.

Location

Knowing the actual location, the system would generate a list of random cities from the actual country. One city is chosen, and from that time on, that city center would be the reference. Given a small change in original location, the fake location would change in the same way referred to the new city. It is possible to choose a city and generate small tracks within the city, simulating a city lifestyle.

This solution presents both versatility and adaptability, providing a new illusory profile that makes deception work. Each app has to be allowed to access the real profile or else the fake profile is presented to this app. This was the solution implemented and it is explained in detail along the next chapter.

Chapter 5

Implementation

The best approach found was explained on the previous chapter, here will be made an exhaustive explanation about the implementation of the proposed solution.

In order to implement the solution achieved, a research was made to evaluate the existent tools that would help to materialize our solution. A very effective tool was found, Xprivacy allows exactly what we need. It feeds applications with data defined previously, it can be fake data or simply no data at all. Xprivacy works as a module of Xposed framework and so, it is important to explain its functionality. Over the next section will be made an exposition of Xposed functioning, referring its strengths to help this study. In the final sections it is explained all the steps taken to achieve our aim. It is explained how the study was implemented to make a new contact list, to randomize identification numbers and finally to schematize a way to give fake locations to applications. Great care was taken to build a solution that was simultaneously coherent and realistic.

5.1 Xposed framework

This framework was created by a XDA user named "rovo89" and it uses root access to allow changes on system level. The main goal of this framework was to make it easier for a developer to change system level features without the need to build a new ROM for the device. These ROM-like features are inserted in individual modules that user can install in the device and make use of Xposed functionality.

The creator of Xposed wanted to make it easy to build the modules, and so, these modules are "written" like other Android apps and just have some additional meta-data.

All starts with a process called Zygote, this is basically the start of every application.

Each application starts as a fork of this process that is started it self by a script "init.rc" when the device is booted. The process is started by "/system/bin/app-process", which loads the required classes and calls the initialization methods. When the Xposed framework is installed an extended "app-process" is copied to "/system/bin". This new process adds an additional jar file to the class-path and calls methods at certain places. It means that the framework has access to Zygote method and can act in its context.

The real power of Xposed is the possibility to intercept method calls. When modifying an APK decompiling it, the developer is changing its behavior by inserting or modifying the code but then, the APK has to be recompiled and signed which is not a versatile solution. With Xposed it is possible the keep APK code unchanged while being able to inject in the execution some code before and even after methods calls. This is useful once we can change every application operation without having to decompile and recompile them. Xposed allows to make changes in applications, modifying any method. Consequently so given a certain call, it is possible to change arguments, change static variables, call other methods than the original, alter the result or even skip that call. Xposed presents a powerful solution to common problems among developers and it is very flexible in the way it permits the modification of any method known.

XPrivacy uses Xposed features to intercept a large number of functions and takes action over applications according with a policy database defined by the user. It feeds applications with fake data or no data when they are restricted to access certain data.

5.2 Contacts

To accomplish the aim of this dissertation, it was necessary to make some changes on Xprivacy module and create some mechanisms to allow the creation of an illusory profile.

The best way found to create a totally new contact list, was using the actual list and shuffle it in some way. The first procedure was to obtain the actual list of contacts by simply query the contacts database as seen on figure 5.1. In this step it was needed to add a read contacts permission to Xprivacy.

The next step was to divide names, surnames and phone numbers into different arrays in order to create new complete contacts, figure 5.2.

After that, using the Fisher-Yates shuffle algorithm, the three arrays were shuffled creating new lists of names, surnames and numbers, figure 5.3. This algorithm was chosen because of its efficiency and unbiased results.

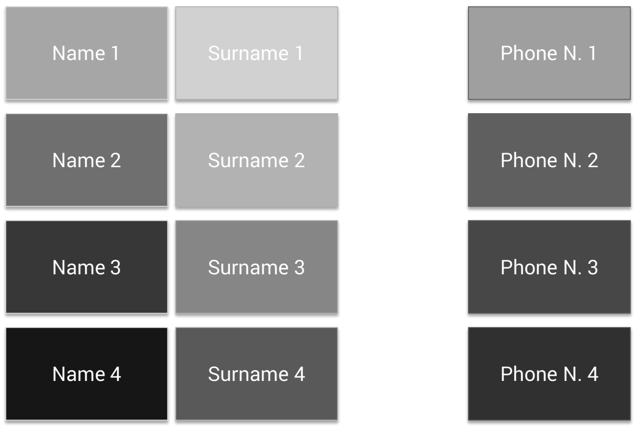

Figure 5.1: Contact list retrieved from contacts database.

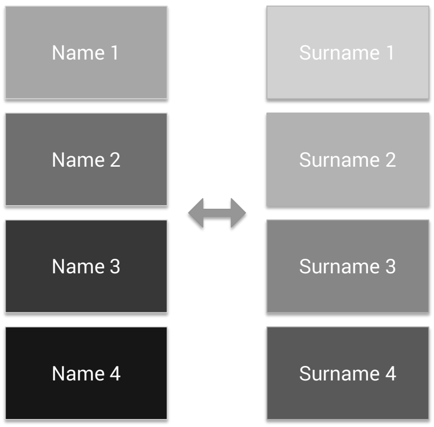

Figure 5.2: Contact list separated into arrays.

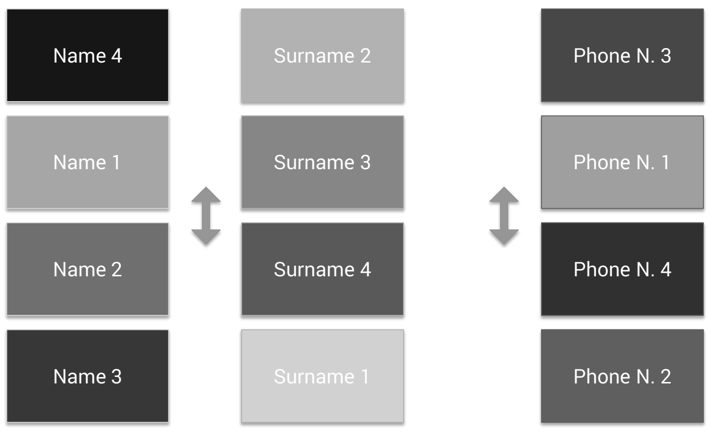

Figure 5.3: Arrays mixed with Fisher-Yates algorithm.

Proceeding with these steps, the arrays of names and surnames were joined back again, building a new contact list, figure 5.4.

The contact phone number was generated by randomizing the last six digits of each number as seen on figure 5.5, there is also the worst case scenario in Portugal. For instance, if a phone number had the country calling code and the operator code these digits wouldn't suffer any change. This solution takes by example the case of Portugal and is very effective once the contact list preserves the original codes.

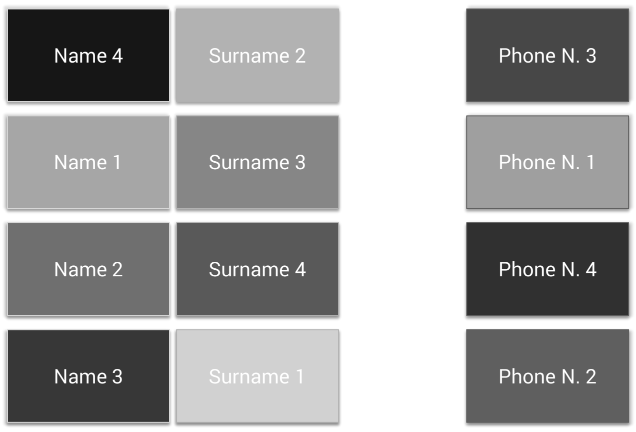

Figure 5.4: New mixed contact list.

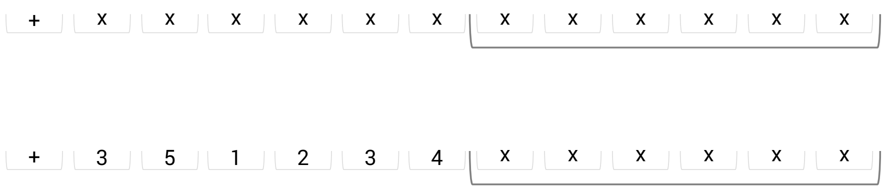

Figure 5.5: Structure of a phone number and the worst case scenario.

As Xprivacy feeds a restricted app with an empty cursor, the way to show the new contact list is to create a new contacts provider based on the contacts provider source code made available by Google with nearly the same functions. From there, was simple to create the all new contact list, every name and surname made one entry adding to it the phone number. As the new contacts provider has restricted features it was not possible to add photo thumbnails to contacts. The figure 5.6 illusrates the change made on Xprivacy to enable the new contacts provider and A restricted application is now redirected by Xprivacy to the new contacts provider and there is no problem if that data is leaked because it is fake data. Each application can access either the original contact list or the fake one according with the restriction list available on XPrivacy.

5.3 Identification

Some identification numbers subject to leakage were identified. A few were likely to change on Xprivacy settings, but again, this isn't a smart randomization, not having in mind the specifications of each identification number.

For this section, were identified the parameters that if leaked, could be a privacy constrain to device's user. That parameters were analyzed in order to decide how data can be randomized. The parameters found were:

- Identification
  - Google advertising ID
  - Device serial number
- Network
  - IP
  - MAC
- Phone
  - IMEI
  - Phone type

Identification

Starting with identification parameters, Google advertising ID as values similar to this,

"38400000-8cf0-11bd-b23e-10b96e40000d"

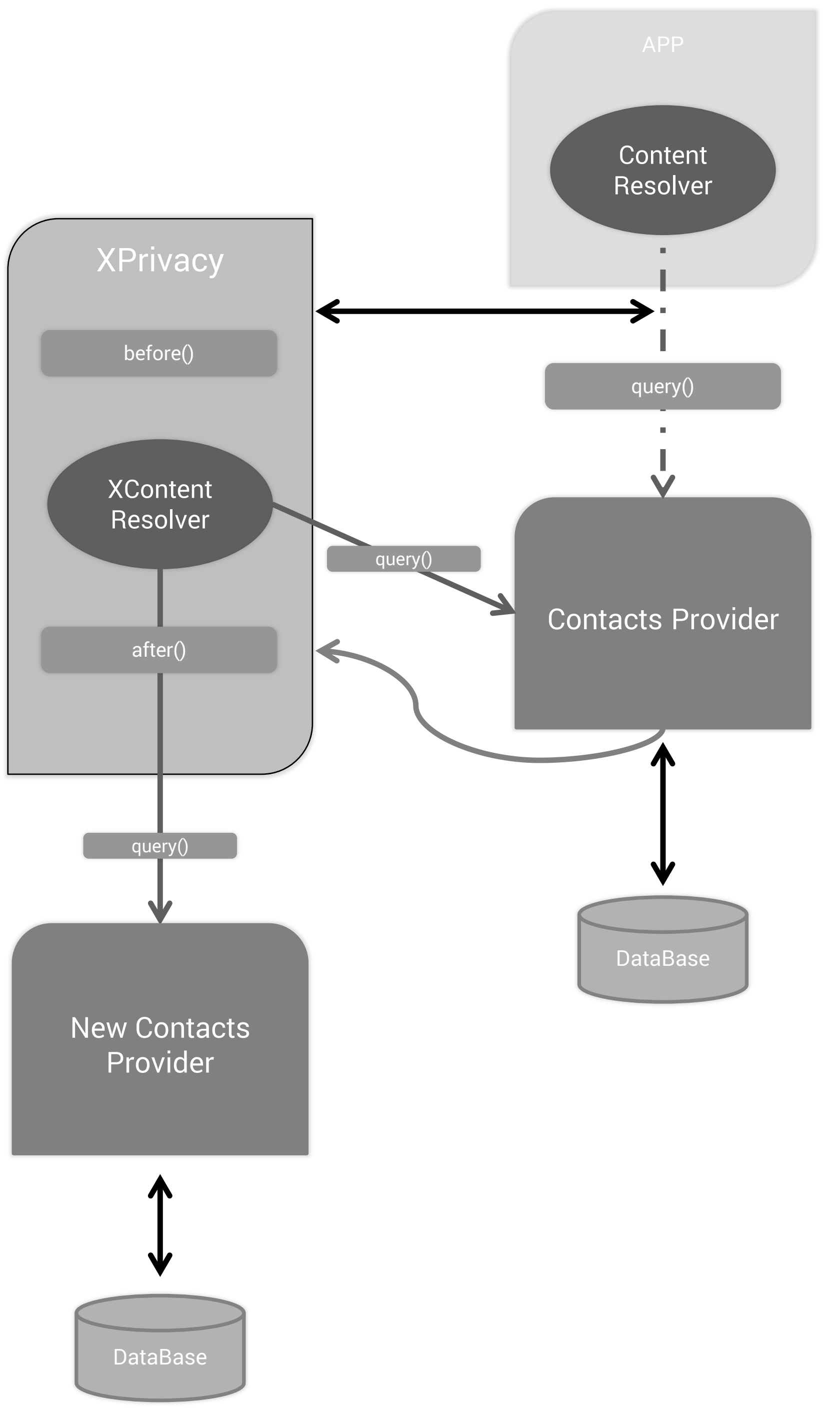

Figure 5.6: The implementation of a new contacts provider on XPrivacy

this identifier has a string format of UUID (Universal Unique Identifier) version 1, compared with MAC adress. There is a Java library "java.util.UUID" that allows to create a new UUID.

In order to deal with the device serial number, as a serial number can differ from different manufacturers, it is necessary to randomize the actual device serial.

Network

The new IP is obtained recurring to the actual IP address and is obtained changing the last values of the address, based on a local network address,

192.168.XXX.XXX

As a MAC address has a UUID format, Xprivacy has a proper function to get a random MAC address.

Phone

Xprivacy has internal functions that are used to do the normal randomization of the IMEI value and calculation of the check digit, following Luhn algorithm. In our solution, IMEI is obtained using the same referred functions.

5.4 Location

On Xprivacy it is possible to define a fake location to feed all applications with location restriction. This location can be arbitrary or defined by user. Our goal was to make a coherent solution based on user habits, making this a more realistic approach. Using the current location, the new Xprivacy module will once find another city within the actual country and establish there a new base. This works like the user having a new home place and he does all the movements from there. For instance, as we see on figure 5.7 if the base city of the user is Aveiro, Xprivacy will choose another city in Portugal, let's say Porto, and from now on Porto will be the base city for all restricted apps. If the user travel around Aveiro city, the restricted apps will see a movement in Porto but always referred to the original movement, figure 5.8.

The implementation of the fake location needed two steps: the first one was to determine the current country and city, and the other one was to search for a city on the same country in order to define the new base reference.

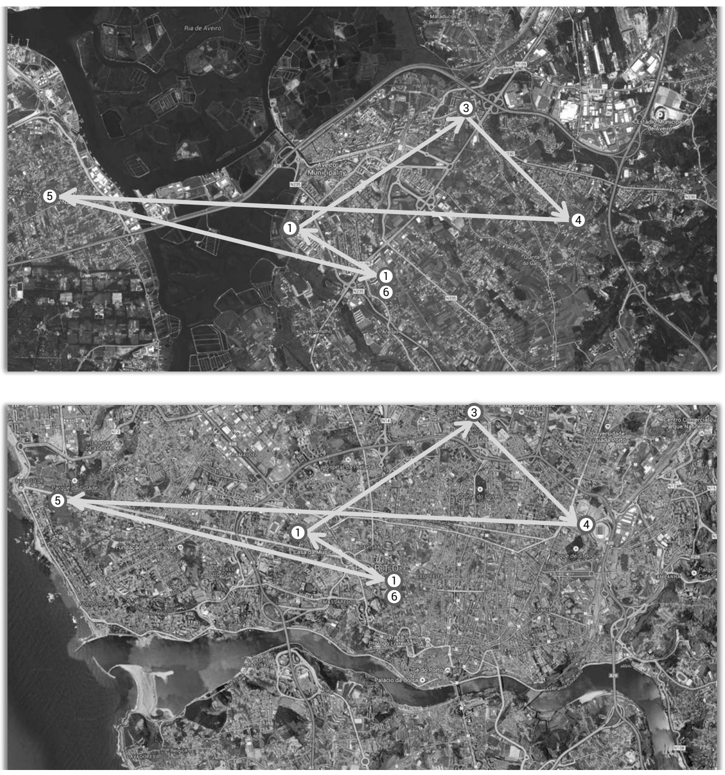

Figure 5.7: Example of location feature.

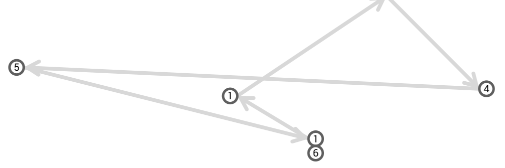

Figure 5.8: The common path for the two cities.

Chapter 6

Future Work

Here are presented some possibilities of future work, based on all information collected and also on the implemented solution.

6.1 Extension of the work

This work could be easily extended to other sensitive data. Some examples will be given about possibilities of extension of this study. There are numerous values that can changed and be part of the illusory profile.

User accounts may be different than original, based on the name of device's owner it is possible to create the illusion of different user accounts.

It is possible to change data such calendar entries where the existing events could be reschedule to a new date and different location.

Even the dictionary can filled with some random words from the actual country. In the same line, browser history can be replaced with some of the most visited websites in the country.

Call log could be replaced by a new one based on the new contacts list on the mobile.

This would allow to create a secure device sharing only the data that user wants to share.

6.2 Data Network

Based on this thesis, each user has its own profile and a fake profile, one possible future work would be to create a network where each user shares his own fake profile. The data each user would be able to share is the created data, the fake one, as an example users could share some of their calendar entries. According with the previous section, these entries would be different than original in time and location. Sharing the fake data such calendar entries, contacts or even location history wouldn't be a major issue for the user as shared data has no meaning for the new user.

With this, a data network could be created, where all users could ask for new data, always being coherent with the actual country or the device's original country.

Chapter 7

Conclusions

Android is nowadays the most used mobile platform, and it is assuming a crescent role in people lifestyle. So, it is very important to provide mechanisms to Android users that ensure the protection of their sensitive private data.

In this document a study was presented about privacy on Android operating system. The main goal was to improve Android privacy using confinement and illusion. An exhaustive research was conducted in order to evaluate the state of the art due to Android privacy leaks and possible improvements. Android operating system was deeply studied as well its security architecture.

For the purpose of achieve the main goal, some solutions were equated and the one that does prevail versatility and at the same time, realism was chosen to implement on Android OS.

It was fundamental the depth knowledge of Xprivacy functions and architecture. It works on top of Xposed framework and takes advantage from its "hook" capabilities on the entire Android system. The use of this tool was crucial to the development of the present thesis, providing tools needed to achieve our proposed solution.

Some proposals to future work were made on Chapter 6, always having in mind the work developed before. Along this thesis, user privacy and a safe digital environment were always the principles. The perfect case is where the user feel safe and doesn't have to think about privacy invasion. Maybe now in general people still feel safe, because they aren't aware of the flaws of digital world and the value of their own data.

There are several studies to identify supposed failures on Android architecture, and some solutions were developed to solve that flaws. However, these improvements are made here and there, what suggests that Android is still an immature OS. And if once, we used phones to nothing more than make and receive calls, today smart-phones are more similar to computers, and almost every computational task can be made on mobile devices, so it is necessary to improve its security and its mechanisms to prevent sensitive private data leakage.

A great effort is being developed by the huge community of Android developers to make this operating system a better option when looking for safeness and safeguard privacy.

Bibliography

[1] Activities | Android Developers, . URL http://developer.android.com/guide/components/activities.html.

[2] Android Security Overview | Android Developers, . URL http://source.android.com/devices/tech/security/.

[3] What is App Ops, and why did Google remove it from Android? | Pocketnow. URL http://pocketnow.com/2013/12/17/app-ops.

[4] Gartner Says Worldwide Tablet Sales Grew 68 Percent in 2013, With Android Capturing 62 Percent of the Market. URL http://www.gartner.com/newsroom/id/2674215.

[5] Nomes e mais nomes: Nomes populares em Portugal - Top 100 de 2013 -. URL http://nomesportugueses.blogspot.pt/2014/01/nomes-populares-em-portugal-top-100-de.html.

[6] <permission> | Android Developers. URL http://developer.android.com/guide/topics/manifest/permission-element.html.

[7] Validating Security-Enhanced Linux in Android | Android Developers. URL http://source.android.com/devices/tech/security/se-linux.html.

[8] Android: A visual history | The Verge. URL http://www.theverge.com/2011/12/7/2585779/android-history.

[9] Patrick P F Chan, Lucas C K Hui, and S.M. Yiu. DroidChecker : Analyzing Android Applications for Capability Leak Categories and Subject Descriptors. pages 125–136.

[10] Lucas Davi, Alexandra Dmitrienko, A R Sadeghi, and Marcel Winandy. Privilege escalation attacks on android. *Information Security*, 2011. URL http://link.springer.com/chapter/10.1007/978-3-642-18178-8_30.

[11] Quang Do, Ben Martini, and Kim-Kwang Raymond Choo. Enhancing User Privacy on Android Mobile Devices via Permissions Removal. *2014 47th Hawaii International Conference on System Sciences*, pages 5070–5079, January 2014. doi: 10.1109/

HICSS.2014.623. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6759226.

[12] William Enck, MacHigar Ongtang, and Patrick McDaniel. Understanding android security, 2009.

[13] Marko Gargenta. *Learning Android.* O'Reilly, 2011. ISBN 9781449390501. URL http://books.google.com/books?hl=en&lr=&id=oMYQz4_BW48C&oi=fnd&pg=PR5&dq=Learning+Android&ots=S21iQbor1l&sig=VoOhmo4yTOdpOVmIXo3kxr7WgDE.

[14] Clint Gibler, Jonathan Crussell, Jeremy Erickson, and Hao Chen. AndroidLeaks: Automatically detecting potential privacy leaks in Android applications on a large scale. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 7344 LNCS, pages 291–307, 2012.

[15] Sheran Gunasekera. *Android Apps Security.* Apress, Berkeley, CA, 2012. ISBN 978-1-4302-4062-4. doi: 10.1007/978-1-4302-4063-1. URL http://www.springerlink.com/index/10.1007/978-1-4302-4063-1.

[16] Ojas Kulkarni. Google goes hard on Malware for Android platform | Gadget Cluster. URL http://www.gadgetcluster.com/2014/04/google-goes-hard-on-malware-for-android-platform/.

[17] Christopher Mann and Artem Starostin. A framework for static detection of privacy leaks in android applications. *Proceedings of the 27th Annual ACM Symposium on Applied Computing - SAC '12*, pages 1457–1462, 2012. doi: 10.1145/2245276.2232009.

[18] Asaf Shabtai, Yuval Fledel, Uri Kanonov, Yuval Elovici, Shlomi Dolev, and Chanan Glezer. Google android: A comprehensive security assessment. *IEEE Security and Privacy*, 8(2):35–44, 2010.

[19] Jeff Six. *Application Security for the Android Platform.* O'Reilly, 2008. ISBN 9781449315078. URL http://medcontent.metapress.com/index/A65RM03P4874243N.pdf.

[20] RJG Vargas, EA Anaya, RG Huerta, and AFM Hernandez. Security controls for Android. *CASoN*, pages 212–216, 2012. URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Security+Controls+for+Android#1.

[21] Zhemin Yang and Min Yang. LeakMiner: Detect Information Leakage on Android with Static Taint Analysis. *2012 Third World Congress on Software Engineering*, pages 101–104, November 2012. doi: 10.1109/WCSE.2012.26. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6394931.

[22] Yajin Zhou, Xinwen Zhang, Xuxian Jiang, and VW Freeh. Taming information-stealing smartphone applications (on android). *Trust and Trustworthy Computing*, (November 2009), 2011. URL `http://link.springer.com/chapter/10.1007/978-3-642-21599-5_7`.