Controlo e Ocultação de dados pessoais em dispositivos móveis

João Miguel Campos Lacerda Marques
Mestrado Segurança Informática
Departamento de Ciência de Computadores
2016

Orientador
Prof. Doutor Manuel Correia, DCC-FCUP

U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/_____/_____

Abstract

Smart-phones became ubiquitous. Their users use them not just to access into the cyberspace, but also to store, search and manipulate any type of personal and professional information.

Nowadays the operational systems of smart-phones don't have the adequate controls that can assure the safety of personal information of their users.

The Operational System Android allows to install applications from a variety of sources available in the official market Google Play [26]. Many of them are free in terms of monetary costs, but implies a cost in terms of user's privacy, since these applications access the private resources even if it is not required.

We can verify in [22] [38] [8] [9], that there are applications available to download in Google Play which transfer the private information of the costumers not just to their servers but also to remote advertising and analytics servers. This is extremely worrying because it creates a threat to the safety and privacy of their users.

Based on this premise, the aim of this work is to design a software component for the Android, which allows the user to customize replies to requests made by applications to sensitive information. Such as: providing fake location, fake device unique identifiers, fake SIM (Subscriber Identity Module) card unique identifiers, among others. This way, creating a fake user profile for each application installed on the device.

Keywords: Android, Privacy, Security.

Resumo

Os dispositivos móveis inteligentes tornaram-se ubíquos. Os seus usuários utilizam-nos não só para aceder ao ciberespaço como para armazenar e consultar qualquer tipo de dados de caráter pessoal ou profissional.

Atualmente, os sistemas operativos destes dispositivos ainda não possuem os controlos adequados de forma a garantirem aos seus utilizadores a segurança necessária da sua informação privada.

O sistema operativo Android permite que o utilizador possa instalar aplicações das mais variadas fontes, disponibilizadas na loja oficial do Android denominado de Google Play [26]. Muitas delas são gratuitas em termos monetários para o utilizador, no entanto, muitas das vezes, implicam custos à privacidade do utilizador, pois acedem a recursos privados do dispositivo mesmo que não necessitem para o seu funcionamento legítimo.

Como podemos verificar em, [22] [38] [8] [9], existem aplicações disponíveis para transferência no Google Play, que transmitem informação do dispositivo não só para os seus próprios servidores mas igualmente para servidores externos de análise de dados e publicidade. Isto é extremamente preocupante, criando assim, uma enorme ameaça à segurança e privacidade do utilizador.

Tendo por base esta premissa, o objetivo deste trabalho é a concepção de um componente de *software* para o sistema operativo Android que permita ao utilizador customizar as respostas aos pedidos efetuados pelas aplicações a informações sensíveis. Como por exemplo, fornecer uma localização diferente da sua, transmitir diferentes identificadores únicos do dispositivo, do cartão SIM (Subscriber Identity Module), entre outras. Conseguindo assim criar um perfil falso para cada uma das aplicações instaladas no dispositivo.

Palavras-chave: Android, Privacidade, Segurança.

Agradecimentos

A todas as pessoas que influenciaram o meu percurso de vida.

Conteúdo

Lista de Tabelas

Lista de Figuras

Acrónimos

BS Base Station

BSN Body Sensor Network

HTTP Hypertext Transfer Protocol

TCP Transmission Control Protocol

UDP User Datagram Protocol

GPS Global Positioning System

OEM Original Equipment Manufacturer

IPC Inter-process Communication

UID User Identifier

GID Group Identifier

MAC Media Access Control

ADB Android Debug Bridge

USB Universal Serial Bus

API Application Programming Interface

ROM Read Only Memory

PID Process Identifier

JVM Java Virtual Machine

PAP Policy Administration Point

PEP Policy Enforcement Points

PDP Policy Decision Point

IMEI International Mobile Equipment Identity

SSL Secure Sockets Layer

A&A Analytics and Advertising

NFC Near Field Communication

MCC Mobile Country Code

MNC Mobile Network Code

LAC Location Area Code

CID Cell Identifier

BSSID Basic Service Set Identifier

SSID Service Set Identifier

URI Uniform Resource Identifier

ICCID Integrated Circuit Card Identifier

IP Internet Protocol

IDE Integrated Development Environment

SIM Subscriber Identity Module

GSM Global System for Mobile Communications

TAG Type Allocation Code

IMSI International Mobile Subscriber Identity

UMTS Universal Mobile Telecommunication System

MSIN Mobile Subscription Identification Number

WLAN Wireless Local Area Network

BTS Base Transceiver Station

IEEE Institute of Electrical and Electronics Engineers

OUI Organizationally Unique Identifier

SMS Short Message Service

MMS Multimedia Messaging Service

JSONs JavaScript Object Notation

Capítulo 1

Introdução

A mobilidade tornou-se imprescindível nas nossas vidas pessoais e profissionais. Atualmente os telemóveis têm um enorme poder computacional o que permitiu aos programadores de *software* explorarem as suas diversas funcionalidades, nas quais se destacam: o acesso à Internet através de redes móveis e redes sem fios; Bluetooth; utilização de sensores; criação e leitura de ficheiros de multimédia; realização de operações bancárias.

Segundo um relatório de estatístico da *Ericsson* [12], em 2014 existiam 2.4 biliões de utilizadores de telefones inteligentes e este número tende a aumentar para 6.1 bilhões até 2020, o que representará cerca de 70% da população global.

A Google, empresa proprietária do telefone inteligente Nexus e responsável pelo desenvolvimento do sistema operativo de código aberto Android, baseado no Kernel do Linux [19], disponibiliza mais de 2.0 milhões de aplicações [48] no Google Play. Ao recuarmos aos telemóveis convencionais, onde existia um número limitado de aplicações, maioritariamente básicas, implementadas pelo seu fabricante e sem acesso à Internet. Estas características tornavam-nos mais seguros relativamente aos telemóveis da última geração, pois estes sistemas estão mais expostos ao mundo no espaço cibernético, originando assim uma maior ameaça à segurança e privacidade dos seus utilizadores.

Mark Zuckerberg, fundador e diretor executivo do Facebook afirmou que a privacidade morreu *"privacy is dead"* [16]. De facto vivemos numa era em que mundo digital comanda grande parte da nossa vida, e, todas as preocupações podem ser poucas. No entanto duvido, que Zuckerberg não esconda o seu PIN (Personal Identifaction Number) do cartão de crédito ou o seu SSN (Social Security Number).

1.1 Descrição do Problema

O funcionamento do Google Play é bastante intuitivo, o utilizador necessita apenas de selecionar a aplicação que deseja instalar, seguidamente é surpreendido pelos diversos requisitos que a

aplicação poderá necessitar para o seu correto funcionamento. Estes requisitos são grupos de permissões [25], para cada um dos grupos tem uma simples e breve descrição do tipo de acesso ao sistema operativo que irá efetuar, ou seja, uma aplicação que necessite de aceder às fotografias necessita de especificar no ficheiro *AndroidManifest.xml* a permissão de acesso *Fotos/Media/Ficheiros*. Precisamente este grupo, permite à aplicação não só aceder às fotografias, mas a todos os ficheiros do dispositivo, incluindo, formatar os dados de memórias externas.

O utilizador tem duas opções: aceita todas exigências da aplicação e posteriormente a mesma é instalada, ou não aceita, e obviamente a aplicação não é instalada.

Uma vez garantidas as permissões e, caso o utilizador pretenda modificar as mesmas, só o consegue se desinstalar a aplicação, obrigando-o a uma revogação automática do contrato [11]. Em suma, não é permitido ao utilizador aceitar apenas algumas das permissões. Não existe forma de restringir o número de vezes ou a hora do dia que a aplicação pode exercer as permissões aceites e por fim, a única maneira de revogar as permissões, uma vez concedidas, é desinstalar a aplicação.

Com o aparecimento da última versão, *Android 6.0 Marshmallow*, modificou para melhor o funcionamento do Google Play e o processo de utilização das permissões pelas aplicações. Com o novo sistema o utilizador, já não é surpreendido com o contrato de permissões antes de instalar uma aplicação, esta é instalada após ser selecionada. Em relação às permissões, são notificadas ao utilizador em tempo real de execução da aplicação, ou seja, quando uma aplicação necessitar utilizar a câmara fotográfica, o utilizador será advertido que a aplicação necessita de acesso à permissão de câmara. Tendo igualmente uma breve descrição do tipo de permissão. Seguidamente, o utilizador ao aceitar, poderá guardar a resposta para não voltar a ser incomodado da próxima vez que seja necessária a permissão para a mesma aplicação. Caso negue o acesso a operação, esta não será efetuada. Todas as permissões das aplicações instaladas podem ser modificadas necessitando, o utilizador de aceder às definições do dispositivo e selecionar o menu *App permissions* como demonstra a Figura 1.1.

Um dos graves problemas é que a maioria dos dispositivo disponíveis com Android são produzidos pelos OEM's (Original Equipment Manufacturer), em parceria com a Google [19]. O que é bastante interessante do ponto de vista dos produtores como a Samsung ou HTC, pois assim conseguem personalizar os seus dispositivos. No entanto, quando a Google desenvolve uma nova versão do sistema operativo, todos os dispositivos Nexus têm disponível a atualização do *firmware*, mas as OEM's não disponibilizam a atualização do sistema operativo para os seus dispositivos, deixando-os vulneráveis a possíveis ataques. É de extrema importância salientar que, o desenvolvimento de novas versões de um sistema operativo, não é só devido à criação de novas funcionalidades, mas igualmente, para tornar o sistema mais seguro.

Segundo os autores de *TaintDroid* [22], um sistema de monitorização do fluxo dos dados privados do utilizador quando são acedidos por aplicações, revela que: foram selecionadas para análise 30 aplicações das mais populares e disponíveis para transferência através do Google Play, requeriam acesso à Internet e a pelo menos uma das seguintes permissões: localização, câmara

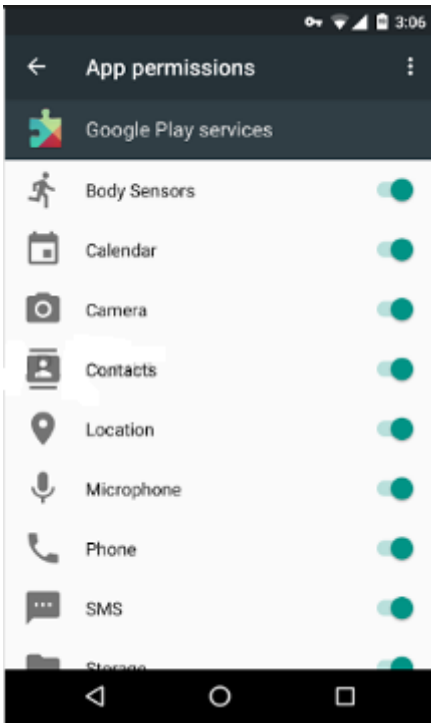


Figura 1.1: Gerir permissões das Aplicações

ou dados do microfone. Ao que concluíram que:

- Existiam 68 instâncias de potencial uso indevido de informações sensíveis do utilizador em 20 das aplicações analisadas.
- 15 das aplicações em análise, transmitiram a informação de localização para servidores de publicidade.
- 9 transferiram o IMEI (International Mobile Equipment Identity) para os seus próprios servidores.
- 2 transferiam igualmente informações do telemóvel para os seus servidores.

O processo de permissões do Android levanta uma série de problemas:

1. A informação acedida por parte das aplicações pode ser transmitidos para servidores externos sem o consentimento do utilizador. Necessitando a aplicação apenas da permissão de acesso à Internet.
2. A descrição das permissões não são suficientemente esclarecedoras para o utilizador final entender os riscos associados.
3. Nas versões anteriores à *Marshmallow*, uma vez aceite o contrato de instalação da aplicação, o utilizador não poderá renegociar o contrato, apenas pode desinstalar a aplicação e desta forma revogar o contrato.

4. Em todas as versões, uma vez dada autorização de acesso à permissão e na versão *Marshmallow* é possível devido ao utilizador poder guardar a suas respostas de acesso às permissões por parte aplicações instaladas no seu dispositivo. Seguidamente as aplicações poderão aceder o número de vezes que pretenderem a essa informação.

1.2 Contribuição para o Problema

Este trabalho contribui com o desenvolvimento de uma aplicação que funcione com uma camada extra de segurança para que, o utilizador tenha o poder de decisão em relação ao uso indevido da sua informação privada, por parte das aplicações instaladas no seu dispositivo.

Com base no estudo efetuado em [38] aprofundado na secção 2.5.3 e no artigo [8], onde foi analisado o comportamento de 2,866 aplicações "confiáveis". Entende-se por confiáveis, aplicações retiradas do Google Play com o fundamento de detetar comportamentos maliciosos de aplicações transferidas da base de dados VirusShare [51]. Observou-se relativamente às aplicações do Google Play, que 68.3 % de todos os acessos a dados sensíveis não resultou numa possível fuga de informação. A fonte de dados mais utilizada pelas aplicações é a sua própria base de dados, seguida da informação do calendário, informação da rede, informação de localização, *content resolver* [5], identificadores únicos, informações de conta, informação dos contactos. O que reflete que grande parte destas aplicações interagem com serviços externos, comunicando informação mantida nas suas próprias base de dados. Em relação às aplicações malignas as fontes de informação mais acedidas são: a informação da rede, seguida da própria base de dados, informação de localização, informação de calendário e identificadores únicos.

Para que o utilizador consiga interagir com a aplicação, e esta faça as modificações necessárias no sistema operativo, foi necessário a utilização da *framework Xposed*, que iremos abordar mais profundamente no capitulo seguinte.

A componente de *software* desenvolvida permite ao utilizador selecionar qualquer aplicação instalada no dispositivo, e, posteriormente decidir que informação sensível deseja precaver da mesma.

Dependendo do tipo de informação que se pretende restringir pode acontecer um dos três casos seguintes: a informação transmitida é a fidedigna do utilizador; a informação transmitida é baseada em dados falsos, e por final, podem ser transmitidos dados vazios ou nulos, que poderá corresponder à impossibilidade de envio dessa informação ou inexistência de dados, como por exemplo, uma lista de contactos vazia.

A solução presente distingue-se pela falsificação inteligente dos dados legítimos do utilizador, conseguindo assim criar a ilusão de um perfil real para cada uma das aplicações instaladas no dispositivo. No processo de transmissão de dados falsos, como por exemplo o IMEI do dispositivo, por cada pedido efetuado por uma determinada aplicação, será gerado um novo.

O utilizador poderá ainda restringir acesso à informação coletada pela Google, como por

exemplo, acesso aos seus contactos e a sua localização atual, em que de igual forma explicado anteriormente, a localização do dispositivo será diferente por cada pedido efetuado pela Google.

1.3 Próximos Capítulos

A restante dissertação está organizada da seguinte forma:

Capítulo 2 Detalhes sobre arquitetura e do modelo de segurança do Android. Análise da *framework* Xposed e por fim, apresentação de sistemas e aplicações com base na segurança da informação privada para sistema operativo Android.

Capítulo 3 Implementação da componente de *software*.

Capítulo 4 Testes efetuados à componente de *software* e resultados obtidos.

Capítulo 5 Termos de contrato e políticas da Google e implicações legais na legislação Portuguesa devido à implementação da componente de *software*.

Capítulo 6 Conclusão e sugestões para trabalho futuro.

Capítulo 2

Estado da Arte

No seguinte subcapítulo iremos abordar sucintamente a arquitetura do Android, seguidamente de uma forma mais aprofundada o seu modelo de segurança. Abordaremos ainda a *framework Xposed* e por final apresentamos sistemas que permitem reforçar a segurança do Android.

2.1 Arquitetura do Android

A arquitetura no Android consiste em quatro principais camadas, que estão representadas na Figura 2.1. Na parte inferior, encontra-se o Kernel que atua como uma camada de abstração entre as aplicações e o *hardware*. No nível superior ao do Kernel encontramos as bibliotecas nativas que permitem ao sistema executar tarefas internas e aos programadores as utilizarem para o desenvolvimento das suas aplicações. Ainda na mesma camada temos a *Dalvik Virtual Machine*, que é uma versão mais leve da *Java Virtual Machine* especialmente desenvolvida e optimizada para o sistema Android, devido aos dispositivos móveis serem menos robustos relativamente aos computadores. A *Application Framework* encontra-se no nível acima das bibliotecas e da máquina virtual e fornece um conjunto de bibliotecas aos programadores escritas em Java e disponíveis através Java APIs. Por final camada do topo encontra-se as aplicações Android com quais os utilizadores interagem.

2.2 Modelo de Segurança do Android

2.2.1 Linux Kernel

Como dito anteriormente o Kernel atua como camada de abstração entre as aplicações e o *hardware* presente no dispositivo [49], assim como em todas as camadas superiores. As principais funcionalidades do Android, como por exemplo, segurança, acesso a redes sem fios e móveis, gerir processos e memória são coordenadas pelo Linux Kernel. Adicionalmente, o Linux kernel permite o isolamento dos processos e um seguro mecanismo de IPC (Inter-process Communication) [41].

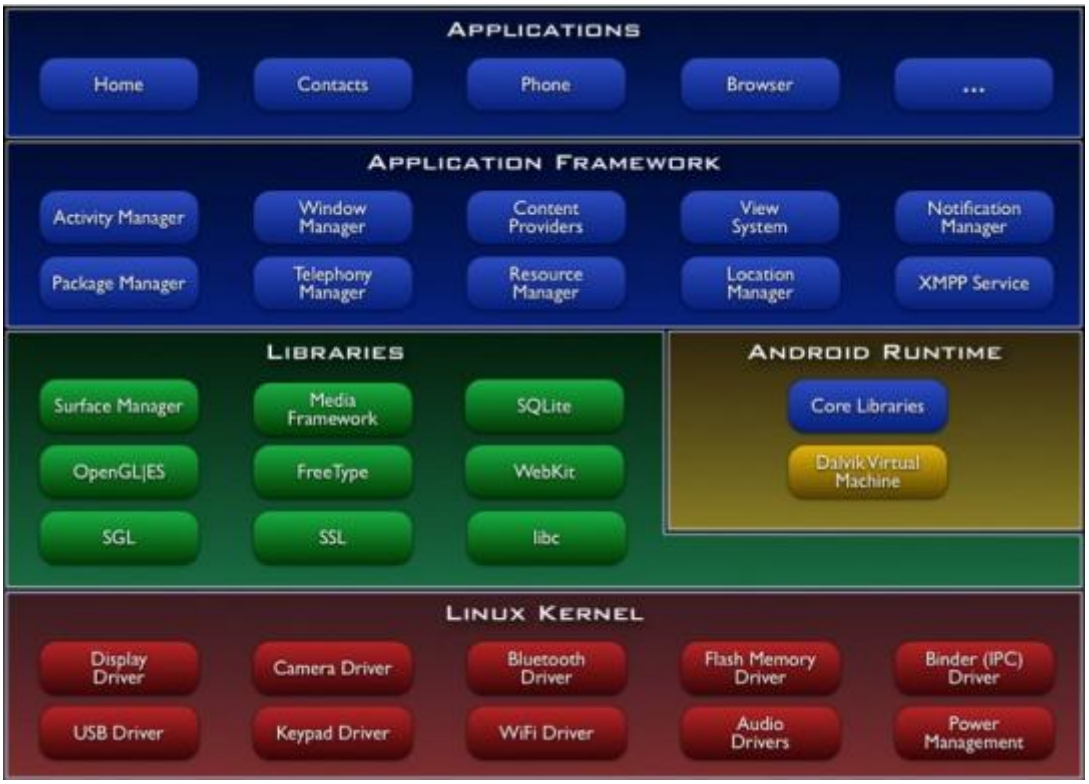


Figura 2.1: Camadas do Android [41]

2.2.2 Android Sandbox

Com a necessidade de isolar as aplicações umas das outras, o sistema operativo Android, fornece um modelo de proteção baseadas no utilizador como no sistema Linux. A cada a cada utilizador é-lhe atribuído UID (User Identifier) e os utilizadores são segregados de modo a que um utilizador não consiga aceder aos dados de outro utilizador [49].

No Android funciona de modo similar, ou seja, a cada aplicação é-lhe atribuído um UID no momento da instalação e a aplicação é executada num processo dedicado, identificado pelo UID atribuído. Quando a aplicação é iniciada, os GID (Group Identifier) e o UID são adicionados ao processo. Se forem adicionadas permissões adicionais, estas são mapeadas para GID's e atribuídas como GID's suplementares ao processo. Prevenindo que uma aplicação maliciosa ao tentar executar uma tarefa maliciosa, apenas o consiga fazer com as suas permissões e desta forma, não consegue comprometer outras componentes ou segmentos do sistema.

O sistema de *sandboxing* é efetuado ao nível do kernel. A segurança entre as aplicações e o sistema a nível de processos é assegurada pelo UID e GID que foram atribuídas às aplicações.

2.2.3 SELinux

Security-Enhanced Linux SELinux (melhoria de segurança do Linux), foi implementada a partir da versão de Android 4.3, com o fundamento das aplicações poderem pedir permissões a nível

do sistema operativo, e os utilizadores poderem garantir ou negar essas permissões. O Android utiliza SELinux de forma a impor mecanismo de segurança MAC (Mandatory Access Control) em todos os processos, incluindo os que funcionam com privilégios de administrador, utilizador *root*. Funciona com o princípio de: por norma negar, ou seja, qualquer coisa que não é explicitamente permitida é negada [49].

2.2.4 Assinatura das aplicações

Todas as aplicações necessitam de estar digitalmente assinadas com um certificado, não necessitando este de ser assinado por uma entidade certificadora, normalmente as aplicações Android utilizam certificados auto-assinados [8] e devem ter validade de pelo menos um ano. A criação das chaves assimétricas é gerada quando o utilizador cria uma conta de programador.

Se o programador desenvolver mais do que uma aplicação e pretender partilhar informação entre as aplicações, basta assinar a nova aplicação com a mesma chave privada permitindo assim que o Android detete que o UID das aplicações são o mesmo.

É de notar que ao utilizar este processo de assinaturas, em mais que uma aplicação, as permissões das aplicações são igualmente partilhadas e que a assinatura das aplicações é uma importante norma de segurança pois, através da assinatura conseguimos obter informação sobre o proprietário da aplicação. Se existir alguma manipulação na aplicação por parte de um utilizador mal intencionado poderá ser detetado, a não ser que a chave privada tenha sido forjada.

2.2.5 Android Debug Brigde

O ADB é uma ferramenta que permite a comunicação direta entre o utilizador e o dispositivo através de uma linha de comandos. Com o aparecimento da versão 4.2.2, a Google reforçou a segurança na existência de uma conexão desta natureza, certificando-se que todas as conexões USB são cifradas e que existe uma confirmação prévia do dono do dispositivo para a utilização da chave assimétrica. A conexão via USB (Universal Serial Bus) e com ADB ligado, pode causar uma enorme ameaça à segurança do dispositivo, por defeito esta propriedade está desativada.

2.3 Modelo de permissões do sistema Android

Existem três tipos diferentes de permissões no sistema Android, permissões API (Application Programming Interface), permissões do sistema de ficheiros e permissões IPC. As permissões API correspondem a mecanismos a nível do Kernel, ou seja, uma aplicação que pretenda aceder a recursos como Internet, sensor de GPS (Global Positioning System), entre outros, necessita de estar inserido no correto GID. Na Figura 2.2 podemos verificar a aplicação do Whatsapp utiliza os grupos de permissões 1006, 1015, 1023, 1028, 3003, que correspondem o acesso à câmara, escrita no cartão de memória, ficheiros de multimédia, leitura no cartão de memória e Internet,

respetivamente. Na Figura 2.3 podemos verificar um excerto de grupos permissões existentes no Android.

```
root@android:/proc/20253 # cat status
cat status
Name:   com.whatsapp
State:  S (sleeping)
Tgid:   20253
Pid:    20253
PPid:   1457
TracerPid:      0
Uid:    10137   10137   10137   10137
Gid:    10137   10137   10137   10137
FDSize: 256
Groups: 1006 1015 1023 1028 3002 3003
```

Figura 2.2: Grupos de permissões da aplicação Whatsapp

```
#define AID_AUDIO         1005  /* audio devices */
#define AID_CAMERA        1006  /* camera devices */
#define AID_LOG           1007  /* log devices */
#define AID_COMPASS       1008  /* compass device */
#define AID_MOUNT         1009  /* mountd socket */
#define AID_WIFI          1010  /* wifi subsystem */
#define AID_ADB           1011  /* android debug bridge (adbd) */
#define AID_INSTALL       1012  /* group for installing packages */
#define AID_MEDIA         1013  /* mediaserver process */
#define AID_DHCP          1014  /* dhcp client */
#define AID_SDCARD_RW     1015  /* external storage write access */
#define AID_VPN           1016  /* vpn system */
#define AID_KEYSTORE      1017  /* keystore subsystem */
#define AID_USB           1018  /* USB devices */
#define AID_DRM           1019  /* DRM server */
#define AID_MDNSR         1020  /* MulticastDNSResponder (service discovery) */
#define AID_GPS           1021  /* GPS daemon */
```

Figura 2.3: Excerto dos grupos de permissões

Como descrito em 2.2.2 as aplicações são executadas em processos diferentes, o que corresponde às permissões do sistema de ficheiros. Os serviços do sistema são executados igualmente em processos separados mas são lhes fornecidos mais privilégios que os processos das aplicações. De forma a ser possível organizar os dados e os sinais entre os processos a *framework* IPC é necessária. No Android isto é possível devido à utilização do mecanismo Binder.

Com a utilização da *framework Binder* é possível a realização de diversas ações, assim como, invocar métodos em objetos remotos como se eles fossem locais, o método que evoca pode ser síncrono ou assíncrono.

Todas as comunicações entre os processos utilizando a *framework Binder* ocorrem sobre o *Linux kernel driver*. Esta componente foi desenvolvida pela Google e posteriormente adicionada ao Kernel.

O diagrama 2.4 demonstra o modelo de comunicação utilizando o Binder. Supondo que a aplicação no Processo 'A' pretende utilizar um serviço que é executado no Processo 'B'. Neste caso o Processo 'A' é um cliente e o Processo 'B' é um serviço.

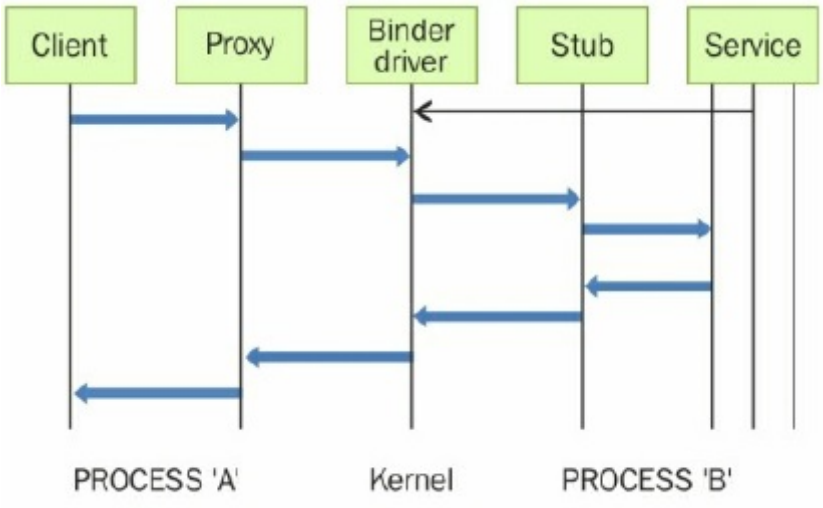


Figura 2.4: Modelo de comunicação do Binder

Todas comunicação entre o cliente e o servidor são efetuadas através de *proxies* do lado do cliente e através de *stubs* do lado do servidor [49]. Portanto, tanto as *proxies* como os *stubs* são responsáveis pelo transmissão e receção dos dados e os comandos são enviados através do *Binder driver*.

2.4 Xposed

Xposed é uma *framework* de código aberto, desenvolvida por *rovo89* [18] da XDA Developers, disponível para versões do Android iguais ou superiores à 4.0.3. A *framework* permite modificar o comportamento dos serviços sistema operativo Android e das aplicações instaladas no dispositivo, sem ser necessário alterar os ficheiros das próprias aplicações. Existe apenas a necessidade de efetuar *root* ao sistema previamente.

Após a sua instalação o utilizador pode adicionar e remover módulos ao Xposed. Sendo necessário ativar o modulo e seguidamente reiniciar o dispositivo para que a *framework* realize as devidas configurações no sistema operativo para a inicialização do modulo.

Tanto os módulos como a própria *framework* são bastante intuitivas para desinstalar. Ao reiniciar o dispositivo todas as configurações originais são repostas, o que é bastante mais prático que instalar ou restaurar uma determinada ROM (Read Only Memory). Outra vantagem é o facto que múltiplos módulos poderem efetuar modificações na mesma parte do sistema ou na mesma aplicação sem que ocorram falhas no sistema operativo.

Para versões do Android iguais ou superiores à versão 5.0 a instalação do *Xposed* não é tão trivial, necessitando atualmente que o utilizador efetue *Flash* ao seu dispositivo e instalar umas das versões do *Xposed* disponíveis em [46]. Existem diversos programas que efetuam estas operações, como por exemplo, o TWRP [50] de código aberto que permite não só instalar uma

nova ROM mas igualmente efetuar cópias de segurança do sistema operativo. O que é bastante benéfico pois se algum processo falhar o utilizador poderá sempre repor a cópia de segurança.

2.4.1 Mecanismo: Zygote

Na fase de arranque do sistema operativo, o processo *init* é o primeiro processo a ser iniciado pelo Kernel PID (Process Identifier) igual a 1. Este, através da análise do *script init.rc* que contém informação dos serviços, ficheiros do sistema e outros parâmetros que são necessários inicializar, inicia-os a todos.

Zygote é um dos primeiros e principais processos a ser criado pelo *init*, devido a funcionar como um *loader* para cada processo na *Dalvik Virtual Machine*. O Android não inicializa um novo processo *Dalvik Virtual Machine* para cada aplicação, o que iria proporcionar uma maior utilização de memória e uma maior complexidade nas suas execuções, em vez disso, é utilizado o processo Zygote. Quando é recebido um pedido para iniciar um novo processo para uma aplicação, faz *fork* de si próprio. Simplificando, a missão do Zygote é iniciar todas as aplicações, sendo o **pai de todos os processos das aplicações**.

No processo de instalação da *framework* é necessário atribuir permissões *root* ao *XposedInstaller.apk* que está disponível para transferência no Google Play. Xposed copia uma extensão do executável *app_process* para o caminho */system/bin* do sistema operativo Android. *App_process* é um programa na linguagem *c++* que é executado no processo de inicialização do sistema operativo resultando no processo Zygote. A extensão do *app_process* permite ao Xposed atuar no contexto do Zygote, em que é adicionado o *XposedBridge.jar* ao *classpath*, o que posteriormente nos permite injetar código antes ou depois de uma chamada a um método do sistema operativo.

O ficheiro *XposedBridge.jar* está localizado em */data/data/de.robv.android.xposed.installer/bin/XposedBridge.jar*, é chamado no início do processo. O seu código pode ser consultado em [45].

2.4.2 Realizar *Hooks* ao Sistema

Entende-se por *hook* um código arbitrário que intersete e manipule chamadas de funções, eventos ou mensagens de um sistema operacional.

Os métodos de *hook* são extremamente robustos devido a permitir ao programador modificar o comportamento uma aplicação sem ser necessário o processo de a descompilar, modificar, recompilar, assinar e executar. Assim, com o sistema baseado em *callbacks*, o programador através de inserção de código consegue efetuar as alterações pretendidas e em memória. No entanto, desenvolver uma solução utilizando o Xposed não é trivial, devido ao programador necessitar de efetuar pesquisa árdua ao código fonte do sistema operativo. Se desejarmos efetuar processo de *hook* para retribuir o IMEI do dispositivo alterado, necessitamos de encontrar todas as classes do sistema que detêm os métodos que nos forneçam essa informação. Por vezes, existem

diferentes classes que invocam diferentes métodos em que os seus resultado são o mesmo. Somente a realizarmos o processo de *hook* em todos as classes e métodos necessários é que podemos garantir que a informação transmitida final é a pretendida.

O código fonte das diversas versões do Android pode ser consultado em [32].

É necessário integrar o XposedBridge.jar no IDE (Integrated Development Environment) para que seja possível aceder às funcionalidades de realizar os *hooks* ao sistema. O ficheiro contém uma função genérica denominada de **hookMethodNative** que é chamada sempre que que exista um processo de *hook* em um método, em vez de ser chamado o método particular.

Ainda no XposedBridge o método **handleHookedMethod** é que irá definir qual a função que irá realizar o *hook*, de acordo com a estrutura global do **hookedMethodCallbacks**, onde pode ser chamado antes ou depois da função para realizar as alterações desejadas como descrito em [44].

2.5 Sistemas e aplicações de proteção informação privada

Nesta seção iremos abordar ROMs e aplicações no sistema Android que permitam ao utilizador proteger a sua informação privada.

2.5.1 Mockdroid

Mockdroid é uma versão modificada do Android 2.2.1 desenvolvida pela Universidade de Cambridge, que permite a simulação dos dados quando as aplicações pretendem aceder a certos recursos do telemóvel. Para cada aplicação instalada no telemóvel o utilizador tem duas opções, ou permite o correto funcionamento da aplicação ou tem a opção de simular dados para os seguintes recursos:

- **Localização**

  Caso o utilizador pretenda simular a informação, as coordenadas de latitude e longitude serão duas constantes , ou reporta que não existe nenhuma solução de localização disponível. Se o acesso à localização for baseada em conexão IP (Internet Protocol) será sempre transmitido *time out*, ou seja, que esgotou o tempo de acesso à rede.

- **Internet**

  Simula que as redes sem fios estão indisponíveis, ou seja, as *sockets* nunca conetam criando um esgotamento de tempo.

- **SMS/MMS/Calendário/Contactos**

  Neste caso a aplicação transmite as bases de dados vazias, simulando que o dispositivo está com os valores de fábrica.

- **ID do dispositivo**

  Simula retorno de um valor constante falso.

- **Broadcast Intents**

  Se a aplicação pedir ou pretender enviar um *broadcast intent* este nunca é enviado ou recebido permitindo assim a prevenção de uma aplicação receber notificações de SMS ou MMS.

No Mockdroid o *Package Manager* original do Android foi modificado e o conjunto de permissões foi duplicado, de forma a ser possível manter a versão original e a versão simulada para cada uma das permissões. Para suportar o controlo de permissões simuladas em tempo de execução, criaram um grupo Unix adicional no sistema, denominado de *mock*, e este grupo tem permissões de leitura e escrita na pasta onde são guardadas as permissões simuladas de todas as aplicações [9].

Existiu a necessidade de utilização de um *inotify* (é um subsistema do Linux Kernel que atua uma extensão do sistema de ficheiros de forma a notificar mudanças no sistema de ficheiros), para que seja possível verificar se existe alteração na pasta das permissões simuladas, em que todas as permissões simuladas são guardadas num ficheiro.

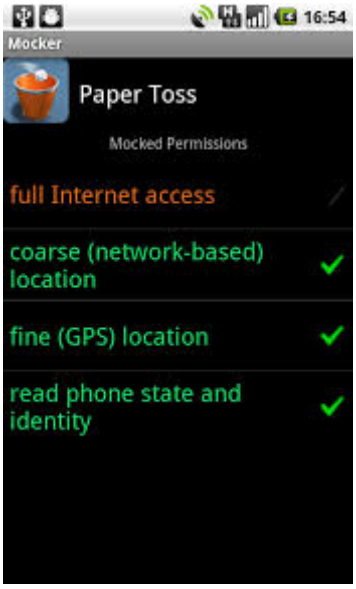


Figura 2.5: Editar permissões da aplicação Paper Toss [9].

As permissões relacionadas com a Internet são tratadas de forma diferente das restantes, pois as conexões IP são realizadas através de chamadas à API pela JVM (Java Virtual Machine) e do Kernel. No processo normal do Android é verificado se o processo que pretende aceder à Internet está inserido no **grupo Inet**. Consequentemente, foi necessário criar um novo grupo **mock_inet**, que basicamente é um espelho do **grupo Inet** para que seja possível controlar, se alguma das permissões das aplicações são necessárias simular. Se o utilizador pretender modificar

a permissão de simulada para o seu valor real é reiniciada a aplicação com as definições de um novo grupo.

Na Figura 2.5 está representado uma parte da interface do utilizador onde é possível editar as permissões requeridas pela aplicação Paper Toss.

2.5.2 TISSA

Versão modificada do Android 2.1 cujo objetivo é prevenir a fuga de informação privada para aplicações desenvolvidas por terceiras entidades. O modelo assume que as aplicações que vêm instaladas de fábrica são confiáveis, pois são realizadas ou inseridas por parte dos fabricantes dos dispositivos. Devido este motivo, não existe a necessidade de serem monitorizadas, relativamente às aplicações instaladas pelo utilizador que são consideradas não confiáveis.

A proteção de informação privada do utilizador é só realizada para o acesso à localização, identidade do dispositivo, contactos e registos de chamadas.

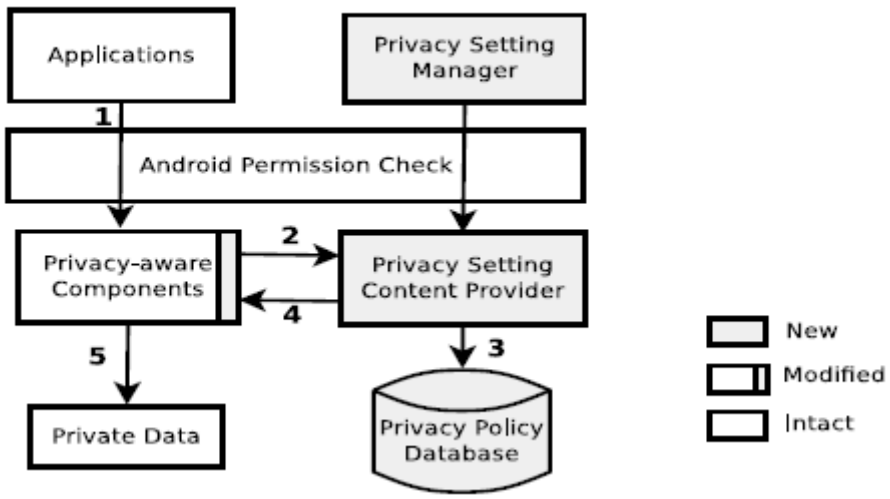


Figura 2.6: Arquitetura do TISSA [52].

Como demonstrado na Figura 2.6, foram criados três novos componentes que estão identificados com uma cor cinzenta:

- ***Privacy Setting Content Provider:***

  É uma PDP (Policy Decision Point) do modelo e é responsável por gerir as definições de segurança das aplicações, permitindo a realização de consultas das definições de privacidade das aplicações instaladas.

- ***Privacy Setting manager:***

  Atua como PAP (Policy Administration Point), onde o utilizador gere as definições de privacidade das aplicações instaladas.

- ***Privacy-aware Components:***

  Componentes funcionam como PEP's (Policy Enforcement Points), onde são incluídos os serviços responsáveis por regular acesso à informação privada do utilizador, tais como, contactos, registo de chamadas, localização e identificação do dispositivo. Ao receberem um pedido por parte de uma aplicação de acesso a um determinado dado privado, os componentes realizam uma consulta ao ***Privacy Setting Content Provider*** para que seja identificado as permissões da aplicação.

Como resultado existe apenas uma instância PDP e PAP enquanto existem múltiplos PEP's, onde são integrados com individuais fornecedores de conteúdos ou serviços na Android Framework.

Quando uma aplicação pretende aceder a dados privados envia um pedido ao correspondente fornecedor do conteúdo, seguidamente é realizada uma consulta às suas definições de privacidade (do fornecedor do conteúdo) para se verificar as definições de segurança da aplicação especificadas pelo utilizador. Estas são guardadas numa base de dados interna *Privacy Police Database* e o resultado é retornado ao fornecedor do conteúdo. Se a leitura da operação é permitida, o fornecedor permite o acesso ao pedido efetuado e retorna o resultado normal para a aplicação. Se a operação não for permitido as definições de privacidade indicam três opções possíveis: *empty*, *anonymized* e *bogus*.

Na opção *empty*, o fornecedor de serviço retorna um resultado vazio à aplicação o que indica que a não existe dados do pedido efetuado. Na opção *anonymized* o fornecedor envia uma versão dos dados originais cifrados com chave simétrica podendo assim a aplicação continuar em execução mas sem acesso à informação privada em texto interpretável. A opção *bogus* retorna dados falsos.

Na Figura 2.7, está representada duas atividades da aplicação, a atividade à (a) lista todas as aplicações instaladas, enquanto a atividade (b) são expostas as quatro definições de privacidade para a aplicação YellowPages.

2.5.3 AppFence

Sistema implementado pela Universidade de Washington e um investigador da Microsoft, onde incluí uma melhoria dos controlos de privacidade para a versão 2.1 do Android. Quando uma aplicação pretende aceder a informações sensíveis do utilizador, e o utilizador não deseja que esses dados sejam acedidos, a AppFence substitui os dados por outros dados ou por dados vazios dependendo das situações. Por exemplo, se a aplicação aceder aos contactos do telemóvel será transmitida uma lista de contactos sem nenhum contacto em contrapartida, se a aplicação aceder à identificação do dispositivo, a aplicação irá receber a *hash* do identificador do dispositivo com *salt*, em que o *salt* é um segredo do dispositivo em conjugação com o nome da aplicação.

AppFence [38] implementa dois controlos complementares denominados de *exfiltration blocking* e *data shadowing* para bloquear a transmissão de dados sensíveis para o exterior através do acesso

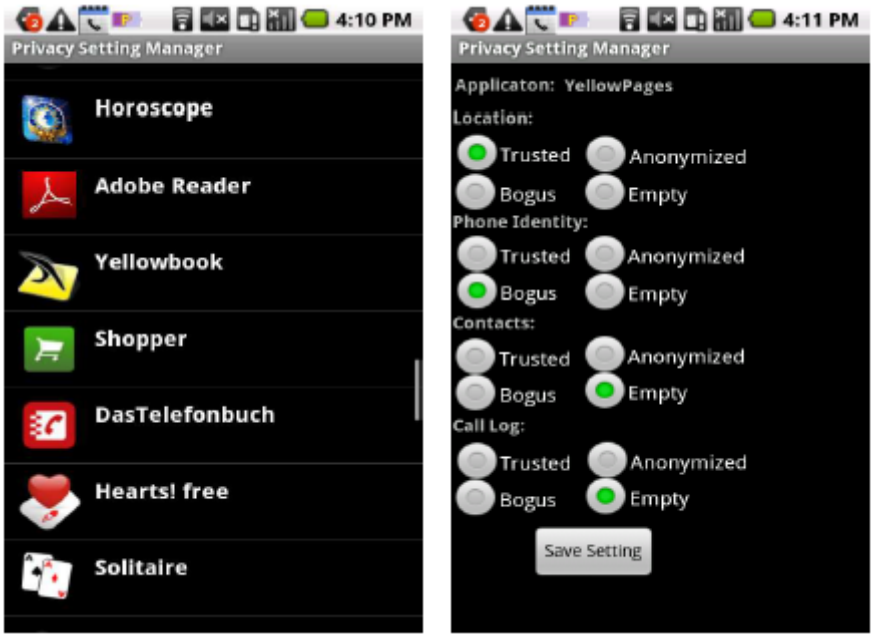


Figura 2.7: **(a)** Lista das aplicações **(b)** Definições de privacidade [52].

à Internet e para prevenir que as aplicações acedam a informação privilegiada. A não ser que necessitem para o seu legitimo funcionamento, respetivamente. Para realizarem o rastreamento que o utilizador considera privada, estenderam TaintDroid *information-flow tracking system* para seguidamente bloquearem o acesso aos dados.

Com a análise de 1100 aplicações Android das mais populares disponíveis no Google Play, selecionando 50 das 22 categorias disponíveis em Novembro de 2010, detetaram que 605 aplicações requeriam acesso a pelo menos um recurso da Figura 2.8 e acesso à Internet. Seguidamente, a fim de realizar uma inspeção mais profunda às aplicações foi utilizado a ferramenta *apktool* do Android para decompor (*disassemble*) e inspecionar os módulos da aplicação de forma a identificarem terceiras entidades de análise de dados e publicidade A&A (Analytics and Advertising), demonstrado na Figura 2.9 .

Das 591 aplicações cerca de (54%) têm um ou mais pacotes A&A incluídos no código em que 361 destas aplicações têm acesso à permissão de Internet, o que possibilita um potencial risco de fuga de informação sensível do utilizador para servidores entidades terceiras.

A fim de identificar quais os tipos de informação sensível transmitida para servidores externos analisaram 110 das 1110 aplicações e detetaram que os dados transmitidos para servidores externos são IMEI, número de telefone, localização, contactos, câmara, conta e microfone.

Demonstrado na Figura 2.10, em 81 aplicações que tinham acesso à permissão do estado do telefone, 31 transmitiram o IMEI para servidores externos desconhecidos e 14 transmitiram para serviços de A&A. Informam ainda que 11 aplicações utilizaram um canal seguro SSL (Secure Sockets Layer) para transmitir tal informação. Em relação à localização, 45 de 73 aplicações com acesso a esta permissão transmitiam a informação para servidores externos desconhecidos

Resource type	Applications
phone_state	374 (34.0%)
location	368 (33.5%)
contacts	105 (9.5%)
camera	84 (7.6%)
account	43 (3.9%)
logs	38 (3.5%)
microphone	32 (2.9%)
SMS messages	24 (2.2%)
history & bookmarks	19 (1.7%)
calendar	9 (0.8%)
subscribed_feeds	2 (0.2%)

Figura 2.8: Permissões mais utilizadas das 1100 mais populares aplicações Android [38].

A&A Module	Applications all 1100		sensitive 605	
admob.android.ads	360	(33%)	225	(37%)
google.ads	242	(22%)	140	(23%)
flurry.android	110	(10%)	88	(15%)
google.android.apps.analytics	91	(8%)	66	(11%)
adwhirl	79	(7%)	67	(11%)
mobclix.android.sdk	58	(5%)	46	(8%)
millennialmedia.android	48	(4%)	47	(8%)
qwapi.adclient.android	39	(3%)	37	(6%)

Figura 2.9: Módulos de publicidade e análise de dados das aplicações em estudo [38].

e 30 transmitiram para serviços de A&A. Relativamente aos contactos, em 29 aplicações que tinham acesso à permissão, 7 transmitiram a informação para servidores externos desconhecidos, no entanto: nenhum foi transmitido para serviços de A&A.

As aplicações Android utilizam ficheiros do sistema para aceder câmara, microfone e registos. Quando as aplicações tentarem abrir estes recursos AppFence fornece a ilusão que está a abrir um ficheiro vazio. Da mesma forma, quando uma aplicação pede acesso aos meta-dados do navegador de Internet, subscrições, contactos, contas e calendário será retornado conjunto vazio de dados. Se a aplicação pedir acesso à localização será retornado as coordenadas 37.421265, -122.084026. Em relação ao número de telefone será transmitido o valor 165063400, relativamente ao IMEI já foi referenciado anteriormente.

De forma a ser possível implementar uma solução com esta finalidade foi necessário modificar o sistema operativo, incluindo as bibliotecas principais do Android, assim como, Android *framework*, conjuntos de serviços e componentes independentes das máquinas virtuais das aplicações. Como podemos verificar a Fiigura 2.11 identifica a arquitetura do AppFence, em que a Dalvik VM

Resource		Demanded	Anywhere		A&A	
phone_state	IMEI	83	31	37%	14	17%
	Phone#	83	5	6%	0	0%
location		73	45	62%	30	41%
contacts		29	7	24%	0	0%
camera		12	1	8%	0	0%
account		11	4	36%	0	0%
logs		10	0	0%	0	0%
microphone		10	1	10%	0	0%
SMS/MMS messages		10	0	0%	0	0%
history&bookmarks		10	0	0%	0	0%
calendar		8	0	0%	0	0%
subscribed_feeds		1	0	0%	0	0%

Figura 2.10: Destino dos dados sensíveis por permissões de aplicação [38].

contém a aplicação e as bibliotecas principais do Android, enquanto que os *Resource Managers* encontram-se na *Android Framework*.

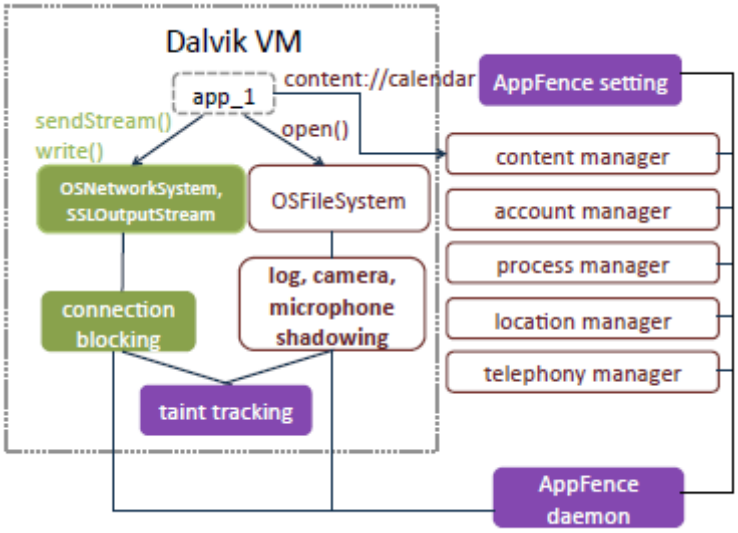


Figura 2.11: Arquitetura do sistema AppFence [38].

Os *Resource Managers* e os componentes do sistema foram modificados para o processo de *shadowing*, enquanto *exfiltration blocking* introduz os novos componentes para bloquear as conexões e rastreamento dos dados sensíveis *(taint tracking)*. O *appFence deamon* executa a sua própria biblioteca e é controlada através da AppFence settings.

Para bloquear a transmissão de informação sensível implementaram uma interseção de todas as chamadas à rede para associar os nomes dos domínios com as *sockets* abertas e para detetar quando os dados sensíveis possam vir a ser transferidos para servidores externos. Quando o

buffer contém os dados *tainted* é descartado e escolhe uma das duas opções: Ou descarta as mensagens indicando que o *buffer* já foi enviado ou emula o comportamento do sistema operativo indicando que está em modo avião e descarta igualmente o *buffer*.

2.5.4 Xprivacy

Módulo para adicionar ao Xposed de código aberto desenvolvido por Marcel Bokhorst (M66B) para a versão do Android 4.0.3 ou superior [40].

O seu foco é prevenir que a informação sensível do utilizador não seja acedida pelas aplicações instaladas pelo utilizador, para esse efeito Xprivacy transmite dados falsos ou dados vazios dependendo do tipo de recurso que esteja a ser acedido pela aplicação instalada. As aplicações que já vêm instaladas de origem com os dispositivos por padrão não são monitorizadas, mas poderão ser se o utilizador o definir nas definições mas não é aconselhado devido a ser mais suscetível ao aparecimento de falhas no sistema operativo.

Para melhor usabilidade e interatividade com o utilizador Xprivacy permite diversas funcionalidades interessantes como, bloquear as permissões todas a uma determinada aplicação apenas com a confirmação de uma caixa verificação, permite o bloqueio através de notificações constantes quando uma aplicação pretender aceder a um recurso, contém ainda informação individual de cada um dos recursos que a aplicação deseja requerer acesso informando o utilizador do perigo de acesso da permissão.

As aplicações podem ser organizadas por vistas de restrições definidas em 24 categorias (*browser*, calendário, contas, chamadas, àrea de transferência *(clipBoard)*, contactos, dicionário, correio eletrónico, identificação, internet, IPC, localização, multimédia, mensagens, rede, NFC (Near Field Communication), notificações, sobreposição de vistas, telefone, sensores, shell, armazenamento, sistema e vistas), ou o utilizador poderá selecionar a opção *all* e todas as restrições são juntas em uma única lista.

Contém ainda símbolos que facilitam a interpretação visual do tipo potenciais riscos de cada aplicação, ou seja, com podemos verificar na Figura 2.12, o símbolo globo informa que uma aplicação tem permissões de uso da Internet, enquanto que o símbolo do triângulo amarelo indica que a aplicação acedeu à permissão recentemente.

Por padrão todas as novas aplicações que forem instaladas não têm acesso a nenhuma categoria, prevenindo assim que a nova aplicação aceda e transmita informação sensível logo após a sua instalação. Xprivacy notifica o utilizador questionando qual o tipo de categorias que a nova aplicação poderá aceder como podemos verificar na Figura 2.13.

Os dados falsos gerados pelo Xprivacy são: informação sobre a conta, Android ID, número de série do dispositivo, identificador *famework* da Google, identificador de publicidade da Google, USB (ID, nome, número), localização, endereços IP, MAC (rede sem fios, bluetooth), BSSID (Basic Service Set Identifier) / SSID (Service Set Identifier), ID de subscrição, IMEI, informações

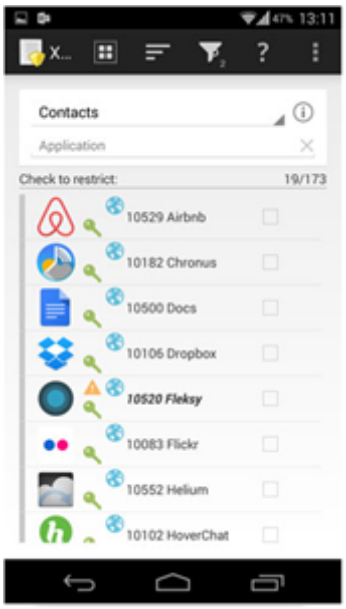


Figura 2.12: Lista de aplicações instaladas e símbolos de alerta

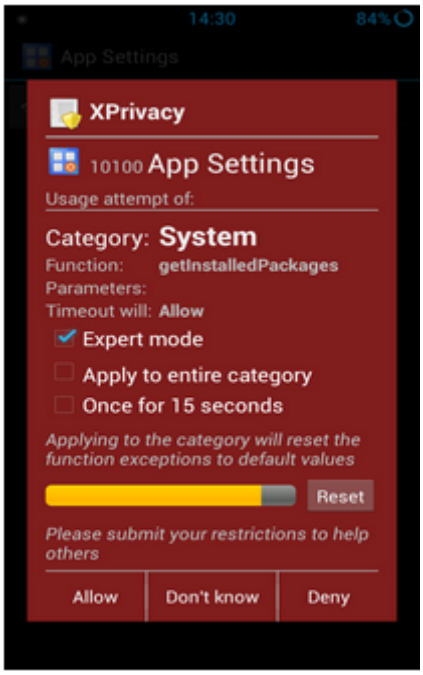


Figura 2.13: Notificação de acesso da aplicação App Settings à categoria System.

da rede sem fios, informações do cartão SIM e por final, o nome e versão do *browser*. A maioria destes campos podem ser introduzidos pelo utilizador, ou serem gerados automaticamente. No entanto, todos eles não têm significado, são estáticos sem intervenção do utilizador e são sempre os mesmos para todas as aplicações. A Figura 2.14 representa Atividade onde o utilizador pode modificar estes parâmetros.

A restante informação que pode igualmente ser protegida, tais como, contactos, calendário,

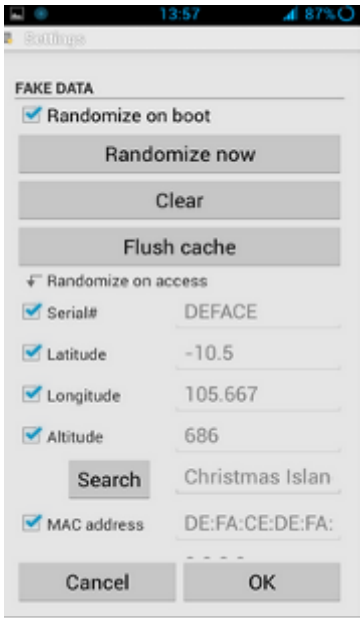


Figura 2.14: Menu de inserção de valores falsos.

sensores, contas do dispositivo, entre outras, são normalmente transferido dados vazios, ou seja, no exemplo de acesso ao calendário, será retornado um calendário como o dispositivo fosse a primeira vez inicializado.

É ainda necessário salientar que Xprivacy tem uma versão profissional contendo mais funcionalidades, tais como, exportar e importar as políticas de proteção definidas na aplicação facilitando assim a necessidade de uma nova configuração, permite ainda restringir o acesso à *Storage*, entre outros. Mas para a sua instalação será necessário a transferência de um donativo. No entanto, o utilizador final necessita de ter algum conhecimento sobre como restringir o acesso ao que pretende, pois muitas das restrições são efetuadas pelo nome dos métodos e outras têm valores por defeito irrealistas.

2.5.5 DonkeyGuard

Desenvolvida pelo criador da *PDroid2.0* um projeto interessante de código livre que já foi descontinuado, permitia igualmente uma inserção de uma camada extra de segurança na informação acedida pelas aplicações. É necessário realizar o *patch* ao sistema tendo em atenção que apenas algumas ROM's é que eram suportadas [14].

DonkeyGuard [13] é um módulo para o Xposed relativamente recente, equiparável ao Xprivacy com menos restrições de dados e não permite o acesso ao código fonte. Ao testar a aplicação vários problemas foram detetados, um deles é a impossibilidade de eliminar as configurações para cada aplicação utilizando a interface gráfica, ou seja, existe a necessidade de as eliminar manualmente podendo ser efetuada através da ligação à ADB.

Um dos problemas é que nem todos os campos necessários para omitir a informação do

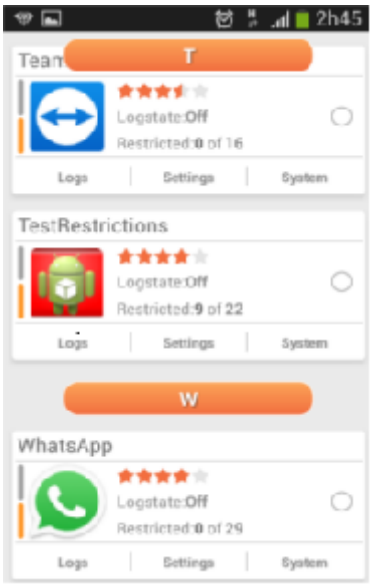


Figura 2.15: Menu de seleção das aplicações.

utilizador são fornecidos, como por exemplo o código da operadora do cartão SIM que incluí o MCC (Mobile Country Code) e o MNC (Mobile Network Code).

Uma boa caraterística de implementação desta aplicação é o facto de estar menos confusa que o Xprivacy e no menu de seleção das aplicações existe uma classificação do nível de perigo de cada aplicação como podemos verificar na Figura 2.15.

Capítulo 3

Implementação

Neste capítulo iremos abordar detalhes da implementação do componente de *software* e as restrições que utilizador poderá impor às aplicações instaladas no dispositivo. As restrições foram selecionadas com base nos artigos [38] [8] devido ao estudo e à análise que ambos efetuaram às diversas aplicações disponíveis no Google Play.

Para a realização da implementação foi utilizado o IDE Eclipse com os *plugins* necessários para programar na plataforma Android.

3.1 Configurações no IDE para utilizar a *framework* Xposed

Para o Xposed detetar a componente de *software* como um módulo é necessário alterar o ficheiro *AndroidManifest.xml*, que está localizado na raiz de todas as aplicações para Android. O ficheiro contém informação essencial sobre a aplicação, como por exemplo, a versão mínima do Android a que se destina a aplicação, as permissões necessárias utilizadas pela aplicação, entre outras. É necessária a adição de três meta-dados ao nodo <application> como demonstra a Figura 3.1.

```
<meta-data
    android:name="xposedmodule"
    android:value="true" />
<meta-data
android:name="xposedminversion"
    android:value="54" />
<meta-data
android:name="xposeddescription"
    android:value="Short description of
the module" />
```

Figura 3.1: Excerto AndroidManifest.xml no nodo <application>

Seguidamente é necessário incluir no caminho *assets* um ficheiro com nome *xposed_init* contendo o nome do *package* seguido do nome da classe principal, neste caso o ficheiro *xposed_init*

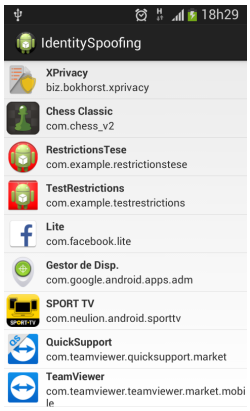


Figura 3.2: Atividade AllAppsActivity.

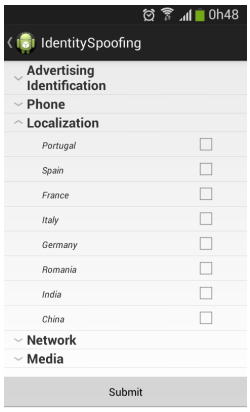


Figura 3.3: Atividade AppRestrictions.

contém *fc.up.identityspoofing.IdentitySpoofing*, em que *fc.up.identitySpoofing* é o nome do *package* e *IdentitySpoofing* representa a classe principal onde são realizados os processos de *hook*.

No ficheiro *XposedBridgeApi.jar* estão incluídas as classes XposedBrigde e XposedHelpers, onde as operações de *hook* são definidas e onde encontramos os métodos de ajuda que permitem ao programador, por exemplo, encontrar as classes e métodos do sistema. É necessário incluir o arquivo *XposedBridgeAPI.jar* na raiz da diretoria da aplicação.

3.2 Estrutura do código e Bases de Dados

Como referido anteriormente IdentitySpoofing.java é a classe principal, onde são inicializados todos os processos de *hook* de todas as classes e métodos. Para que o utilizador consiga interagir com o sistema através de uma interface gráfica são utilizados as Atividades, existem apenas três Atividades no IdentitySpoofing, uma que lista todas as aplicações instaladas no dispositivo que corresponde à classe AllAppsActivity.java 3.2. A classe AppRestrictionsActivity.java fornece ao utilizador uma vista por grupos de permissões para uma determinada aplicação, de forma ao utilizador conseguir selecionar o que deseja restringir. E por último, a classe ActivitySettings.java que permite ao utilizador manipular a localização dispositivo, restringir acesso a contactos, marcadores do navegador de Internet e acesso à informação do calendário por padrão ao iniciar o dispositivo, como podemos verificar na Figura 3.4. É de extrema importância realçar que as definições nesta Atividade permite ao utilizador proteger-se relativamente à informação acedida pelos serviços da Google.

Na Atividade AllRestrictions.java existem cinco grupos distintos *Advertising Identification*, *Phone*, *Localization*, *Network* e *Media* como podermos verificar na Figura 3.3 e abordaremos com mais detalhe cada um dos grupos nas seguintes subseções.

O IdentitySpoofing é composto por três base de dados: IdentitySpoofing.db, Fakedata.db e Usage.db. A **IdentitySpoofing.db** está relacionada com a iteração entre o utilizador e a aplicação, ou seja, tem informação das restrições que o utilizador pretende efetuar nas aplicações

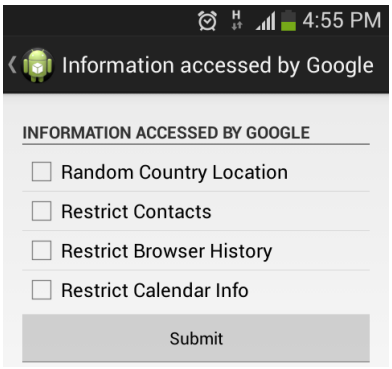


Figura 3.4: Atividade ActivitySettings.

instaladas no dispositivo. É composta pela tabela *restrictions*, que é constituída pelos campos: ***uid, method, package, restricted***, onde ***uid*** e ***package*** representam o UID da aplicação e nome do pacote da aplicação que pretendemos efetuar alterações, respetivamente. Relativamente ao campo ***method*** corresponde ao método do sistema operativo que pretendemos realizar o processo de *hook*. Por final o campo ***restricted*** é um booleano que indica se para uma determinado UID e para o determinado método é ou não efetuado o processo de *hook*.

No que diz respeito à base de dados **Fakedata.db** é composta por duas tabelas distintas de dados falsos ambas com coordenadas de localização de 8 países (Portugal, Espanha, França, Alemanha, Itália, Roménia, Índia e China). A tabela *gps* contém cerca de 1000 coordenadas latitude e longitude de cada um dos países, retiradas do site [24]. A tabela *towersLocation* contém cerca 6000 posições de antenas de redes móveis para cada um dos países. Composta pelos campos ***mcc*** que representa o código do pais da rede móvel. O campo ***mnc*** que representa o código da operadora do serviço móvel. O campo ***lac*** que representa o código da área onde está localizada a antena. O campo ***cellId*** que é utilizado para identificar a antena. E por final, o campo ***cc*** que representa o indicativo do país.

Toda a informação relativa à tabela *towersLocation* excluindo o campo ***cc*** foi retirada do maior projeto do colaborativo de informações relativas às BTS (Base Transceiver Station) e pode ser encontrado em [21].

Relativamente à base de dados intitulada de **Usage.db** é necessária devido à necessidade de guardar informação sobre a interação das aplicações com os recursos do dispositivo. Os campos que a constituem são os mesmos que os da base de dados IdentitySpoofing, mais quatro, ***grouprestriction***, ***time***, ***value*** e ***extra***. O campo ***grouprestriction*** indica o nome do grupo a que a restrição pertence. O campo ***time*** representa o tempo em milissegundos de quando foi efetuado o último acesso ao recurso por parte de uma aplicação. O campo ***value*** e ***extra*** são informações adicionais que podem ser necessárias para contextualizar melhor o tipo de acesso, estes dois campos não são obrigatórios e podem representar informações diferentes para diferentes tipos de acessos a recursos do dispositivo.

3.3 Grupo de restrição *Advertising Identification*

A publicidade nas aplicações descarregadas gratuitamente no Google Play são uma constante, normalmente este tipo de serviços são realizadas por entidades terceiras, proporcionando ao proprietário uma forma de rentabilizar a sua aplicação. Através da utilização de números identificação únicos do Android é possível identificar as preferências do utilizador, posteriormente com base nestas informações permite às organizações efetuarem publicidade direcionada.

Os identificadores únicos de publicidade em dispositivos móveis têm sofrido alterações ao longo dos anos, até à versão 4.4 do Android o *AndroidID* era o único identificador único disponível para fins de publicidade, no entanto devido a preocupações de privacidade do utilizador a Google implementou um novo identificador único denominado de *AdvertisingID*. Atualmente, com o novo identificador único permite ao utilizador selecionar se deseja ou não receber anúncios personalizados e gerar um novo sempre que pretenda. Enquanto que na utilização do identificador persistente *AndroidID* o código é gerado na primeira inicialização do sistema operativo e apenas é possível alterá-lo se utilizador repor as definições de origem do dispositivo [3]. No entanto, os dispositivos Android pelo menos até à versão 5.1 ainda são compostos pelo identificador *AndroidID* e este pode ser utilizado para monitorizar as preferências dos utilizadores.

Todos os processos de *hook* podem ser consultados na classe principal IdentitySpoofing.java e as classes e métodos de cada processo de *hook* relativos a este grupo pode ser consultado na Tabela 3.1.

3.3.1 AndroidID

Um número hexadecimal de 64 bits representado por uma *string*, o utilizador ao selecionar esta restrição será gerado uma nova *string* por cada chamada efetuada por uma aplicação.

3.3.2 AdvertisingID

Com Advertising ID o utilizador consegue desativar os anúncios personalizados para isso terá de aceder à aplicação Definições Google, no entanto os anúncios não desaparecem. Apenas não são efetuados com base na informação histórica de preferências do utilizador, está opção pode ser alterada automaticamente para a sua definição de origem se o utilizador limpar a cache do dispositivo.

O seu formato é um número hexadecimal de 128 bits representado por uma *string*, o utilizador ao selecionar esta restrição será gerado um novo identificador de publicidade utilizando a biblioteca do Java "java.util.UUID".

Nome	Método	Classe
Android ID	getString	android.provider.Settings.Secure
Advertising ID	execTransact	android.os.Binder

Tabela 3.1: Classes e métodos onde são efetuados os processos de *hook* para o grupo *Advertising*.

3.4 Grupo de restrição *Phone*

No grupo *Phone* é possível restringir o acesso indevido das aplicações a informações importantes que identificam o dispositivo, como veremos nas seguintes subseções grande parte destes atributos não são necessários para o correto funcionamento das aplicações. Segundo os estudos efetuados por [38] [8] existe uma grande percentagem de acesso e transmissão da informação do telefone para servidores externos e sem o consentimento do utilizador.

Todos os processos de *hook* podem ser consultados na classe principal IdentitySpoofing.java e as classes e métodos de cada processo de *hook* relativos a este pode ser consultado na Tabela 3.2.

3.4.1 IMEI

Com a implementação da tecnologia GSM nos quatro continentes, mas maioritariamente na Europa, o IMEI permite a identificação individual de cada telemóvel através de um número único, como se fosse o identificador do chassi de um carro ou um endereço MAC de uma placa de rede. Os seus principais objetivos para além da identificação do dispositivo é permitir que seja possível implementar medidas contra equipamentos que tenham sido furtados [7], permite a possibilidade de realizar escutas para detetar operações fraudulentas e a capacidade de repor as definições de origem do equipamento remotamente.

Constituído por 15 dígitos decimais que incluem informação de origem, modelo e número de série do dispositivo. Os 8 primeiros dígitos representam a TAG que identifica modelo do dispositivo móvel, os seguintes 6 dígitos são definidos pelo fabricante do dispositivo e por final o último dígito é opcional que representa o *checksum* do algoritmo de Luhn [2] como pode ser verificado na Figura 3.5.

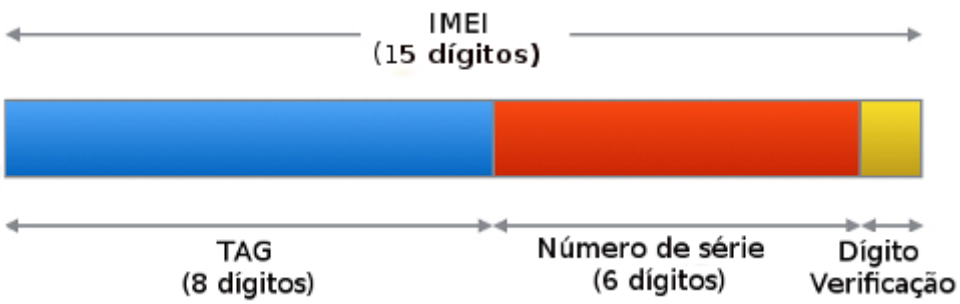


Figura 3.5: Estrutura do IMEI

Existem quatro formas diferentes de um programador ter acesso ao IMEI do dispositivo e estão igualmente representadas na tabela 3.2. Logicamente teremos de realizar o processo de *hook* para cada uma das formas, pois não sabemos qual dos métodos será utilizado pelas aplicações.

Por cada pedido das aplicações ao IMEI será gerado um número diferente, o algoritmo utilizado para gerar e validar o IMEI foi com base na informação disponivel em [39], se o IMEI não passar no teste de validação Luhn será gerado um novo. Todo este processo pode ser consultado no ficheiro XImei.java.

3.4.2 *Device Serial Number*

O número de série do dispositivo em norma é utilizado para identificar dispositivos que não têm a função de telefone, como por exemplo, os *tablets*. No entanto, os dispositivos com as funcionalidades de telefone também podem ser compostos por este identificador único.

O utilizador ao selecionar esta opção serão eliminados os últimos cinco dígitos em hexadecimal do número original e seguidamente geramos os cinco dígitos no mesmo formato e adicionamos ao restante número. O que nos permite transmitir um valor diferente do original sem perder a sua composição e o seu significado.

3.4.3 *Phone Number*

O número de telefone é das informações do utilizador que normalmente é desnecessária para o correto funcionamento das aplicações, este se necessário deveria ser pedido no momento de instalação da aplicação ou na inicialização da aplicação. Os utilizadores comuns não têm a noção de quando uma aplicação tem acesso à permissão de READ_PHONE_STATE que através de código é possível aceder ao seu número de telefone, mas nem em todos os dispositivos. Quando o utilizador ao ativar esta opção será gerado um número aleatório de 11 dígitos iniciado por 00 que representa o sinal + do indicativo do país.

3.4.4 *Hide Running Processes*

A razão de inserir esta restrição é pelo simples facto que aplicações maliciosas ao aceder aos processos que estão em execução podem detetar alguma vulnerabilidade já conhecida e conseguirem efetuar alguma operação maliciosa, desta forma, quando uma aplicação efetuar este pedido será transmitido uma lista vazia de processos em execução.

3.4.5 *Hide Installed Applications*

As aplicações instadas conseguem listas todas as aplicações que existem no dispositivo mesmo sem nenhuma permissão requerida. Do ponto de vista de uma aplicação com intenções maliciosas

Nome	**Método**	**Classe**
IMEI	getDeviceId	android.telephony.TelephonyManager
IMEI	getDeviceId	com.android.internal.telephony.gsm.GSMPhone
IMEI	getDeviceId	com.android.internal.telephony.PhoneSubInfo
IMEI	getDeviceId	com.android.internal.telephony.PhoneProxy
Device Serial Number	get	android.os.SystemProperties
Phone Number	getLine1Number	android.telephony.TelephonyManager
Hide Running Processes	getRunningAppProcesses	android.app.ActivityManager
Hide Installed Aplications	getInstalledPackages	android.app.ApplicationPackageManager

Tabela 3.2: Classes e métodos onde são efetuados os processos de *hook* no grupo *Phone*.

esta operação pode ser o ponto de entrada para detetar se existe alguma aplicação que já tenha sido detetada alguma vulnerabilidade e caso seja afirmativo o atacante poderá tirar vantagem desta situação e posteriormente efetuar um ataque. Quando o utilizador selecionar esta opção nenhuma informação sobre as aplicações será transmitida.

Nas seguintes cinco subseções o acesso à informação privilegiada do utilizador é efetuada através de componentes que fornecem de uma forma segura e robusta os conteúdos a todas as aplicações. Estes componentes permitem às aplicações comunicarem entre si ou que utilizem dados que são partilhados por todo sistema, como por exemplo, os contactos do dispositivo, SMS (Short Message Service), MMS (Multimedia Messaging Service), registo de chamadas, etc.

Para aceder a um Fornecedor de Conteúdos é necessário [4]:

1. Indicar no *AndroidManifest.xml* o Fornecedor de Conteúdos que pretendemos aceder e as permissões necessárias para as operações.
2. Indicar qual URI (Uniform Resource Identifier) do Fornecedor de Conteúdos que é constituído pelas seguintes partes:
   - **Prefixo padrão:** Determina o tipo de recurso ou protocolo, por exemplo, quando acedemos *Word Wide Web* o protocolo é http ou https no entanto os Fornecedores de Conteúdos, iniciamos sempre por *content*.
   - **Autoridade:** Determina o recurso a que desejamos aceder, como por exemplo, as SMS seria "content://sms", em que “sms” representa a autoridade.
   - **Caminho dos dados:** Especifica a parte do recurso a que se quer aceder, como por exemplo, mensagens de entrada o URI ficaria “content://sms/inbox”, em que “*inbox*” representa o caminho dos dados.
   - **Identificador:** Indica individualmente o recurso que se pretende aceder, como por exemplo, “content://sms/inbox/7” aponta para mensagem número 6 guardada na tabela “*inbox*” do Fornecedor de Conteúdos de mensagens.

3. Para aceder aos dados de um determinado Fornecedor de Conteúdos é necessário instanciar um objeto da classe *ContentResolver* [5]. O *ContentResolver* fornece métodos obter dados (*query()*) e para editar dados (*insert(), update(), delete()*. Quando um pedido é efetuado um pedido o*ContentResolver* primeiro verifica se a aplicação tem privilégios necessários para aceder ao recurso analisando o *AndroidManifest.xml*. Em caso afirmativo, envia o pedido para o Fornecedor de Conteúdos respetivo.

   Na Figura 3.6 demonstra três aplicações, cada uma com o seu *ContentResolver* e através de um URI conseguem obter informação de um Fornecedor de Conteúdos em que os dados estão guardados em uma base de dados SQLite.

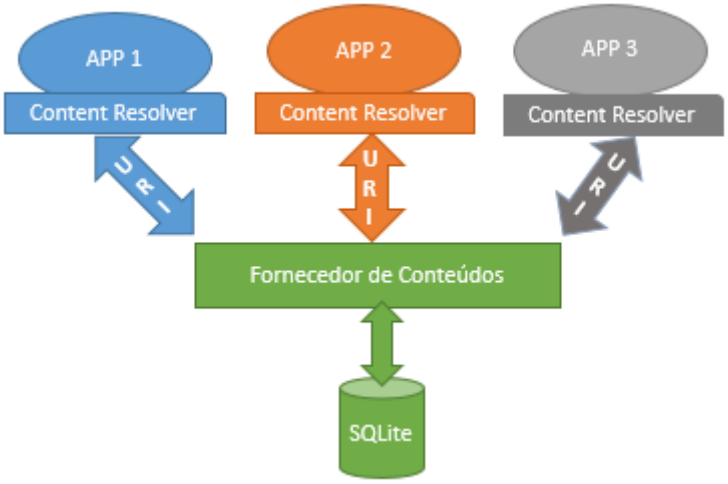


Figura 3.6: Exemplo básico de pedidos ao Fornecedor de Conteúdos

3.4.6 *Block Access Contacts*

Atualmente, a lista telefónica revela bastante informação sensível das pessoas, tais como, morada de casa e do trabalho, número pessoal e empresarial, correio eletrónico pessoal e profissional, data de nascimento, entre outras. Eventualmente existem aplicações que necessitam de acesso à lista telefónica para o seu funcionamento legítimo, no entanto poderá existir aplicações que não necessitam do acesso à lista telefónica para o seu funcionamento e acedem igualmente a todo este conjunto de informações.

Caso o utilizador pretender bloquear o acesso aos contactos serão intersetados os pedidos ao Fornecedor de Conteúdos de contactos com URI iniciados por:

- content://contacts/people
- content://com.android.contacts/profile
- content://com.android.contacts/data
- content://com.android.contacts/contacts

E seguidamente simulamos e transmitimos uma lista telefónica sem nenhum contato disponível.

3.4.7 *Return No Call Logs*

O registo de chamadas poderá igualmente ser uma boa tática para aceder informação relativamente a contactos e costumes do utilizador, ao selecionar esta opção as aplicações não receberam nenhuma informação sobre registos de chamadas do utilizador o URI intersetado tem de ser iniciado por "content://call_log".

3.4.8 *Return No Calendar Info*

Ao utilizar o calendário o utilizador revela imenso da sua vida diária, o acesso a esta informação pode revelar informações relativamente ao seu passado, presente e futuro. Na criação de um registo o utilizador pode adicionar meta-dados, como imagens, localização, descrição, prioridade, entre outras, podendo estas informações serem de caráter pessoal ou profissional. O acesso endivido a este tipo de informação privilegiada do utilizador causa um enorme abuso, pois as pessoas têm o direito a reservar a sua vida, os diferentes aspetos da sua vida do escrutínio de outros. O utilizador ao selecionar esta opção será transmitido um calendário sem registos, o URI intersetado ao Fornecedor de Conteúdos tem de se iniciado por "content://com.android.calendar".

3.4.9 *Return Empty Browser History & Bookmarks*

O utilizador através desta opção pode restringir o acesso ao histórico de pesquisas e aos seus marcadores do navegador de Internet, transmitindo nenhum registo de pesquisas e marcadores. O URI intersetado ao Fornecedor de Conteúdos tem de se iniciado por "content://browser".

3.4.10 *Return Empty SMS & MMS Boxes*

De igual forma, ao selecionar esta opção será transmitido nenhum registo de SMS e MMS na caixa de entrada e de saída do dispositivo. Os URI's intersetados são "content://sms" e "content://mms", respetivamente.

3.5 Grupo de restrição *Network*

Como referido em [8] uma das fontes de dados mais acedidas pelas aplicações são as informações de rede. O Android permite o acesso a informação bastante detalhada sobre as redes já configuradas no dispositivo. Nas seguintes subseções iremos abordar as propriedades que o utilizador poderá restringir quando uma aplicação acede a este tipo de informação.

Todos os processos de *hook* podem ser consultados na classe principal IdentitySpoofing.java e as classes e métodos de cada processo de *hook* pode ser consultado na Tabela 3.3.

3.5.1 MAC *WLAN Inferface*

O endereço MAC é um endereço físico guardado em *hardware* nas tecnologias de rede, tais como, placas rede e Bluetooth, normalmente utilizado para identificação do computador e para controle de acessos em redes de computadores.

O seu formato é um conjunto de 6 bytes, em que cada byte é representado por dois algarismos em hexadecimal seguido do separador (:), os 12 algarismos que formam o endereço correspondem ao total de 48 bits, o que corresponde 2^{48} ou 281474976710656 endereços possíveis.

Os primeiros três bytes são fornecidos pelo IEEE (Institute of Electrical and Electronics Engineers) e têm como função a identificação do fabricante. Os últimos 3 bytes são fornecidos pelo fabricante de forma a identificarem individualmente cada uma das placas. Na Figura 3.7 está representado um exemplo de endereço MAC e a sua estrutura.

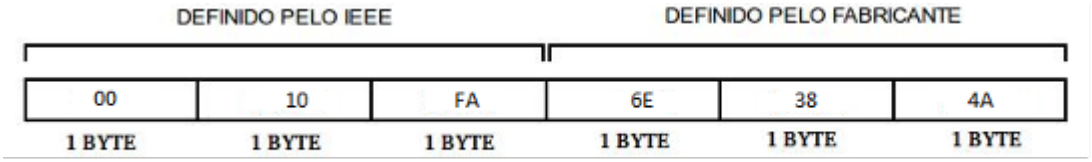


Figura 3.7: Exemplo e estrutura do endereço MAC

Quando o utilizador ativa esta opção o endereço MAC da placa de rede sem fios serão apenas gerados os últimos 2 bytes que representam 2^{16} diferentes possibilidades. Desta forma, conseguimos garantir que os 3 bytes definidos pelo IEEE e o primeiro byte definido pelo fabricante estão corretos, o que torna o novo endereço MAC mais realista em comparação ao gera-lo na sua totalidade sem significado em todos bytes que o compõem.

3.5.2 SSID

Conjunto até 32 carateres utilizados para identificarem uma rede sem fios 802.11, ao que normalmente intitulamos de “nome da rede”. O seu propósito é devido à necessidade de múltiplas redes sem fios poderem ser administradas pelo mesmo router e desta forma é possível a separação e identificação de cada uma das distintas redes sem fios. Ao selecionar esta opção o será transmitido um nome de rede aleatório de 6 a 15 carateres em cada pedido efetuado.

3.5.3 BSSID

Representa o endereço MAC do ponto de acesso à rede sem fios, o utilizador ao ativar esta opção o IdentitySpoofing irá gerar um novo endereço MAC de igual método ao efetuado em 3.5.1.

3.5.4 *Block List of APs found*

O Android tem uma função que permite às aplicações aceder a informações sobre os pontos de acesso à rede que foram detetados na última pesquisa. Informações como: qualidade do sinal, detalhes sobre autenticação e os tipos de cifra suportados pelo ponto de acesso, a frequência da rede, largura de banda e o *timestamp*. Para efetuar esta operação são necessárias duas permissões ACCESS_WIFI_STATE e CHANGE_WIFI_STATE. O utilizador ao selecionar esta opção será transmitido uma lista vazia.

3.5.5 *Block List of Configured Networks*

Através da permissão de ACCESS_NETWORK_STATE as aplicações podem aceder a uma lista com informações detalhadas das redes que já tenham sido acedidas pelo dispositivo. No conjunto de informações destacam-se o SSID, BSSID, os protocolos permitidos, os algoritmos de autenticação permitidos, cifras permitidas para trafego *unicast*, *multicast* e *broadcast*. Ao selecionar esta opção não será transmitida nenhuma informação sobre as redes configuradas no dispositivo sendo retornado “null”, que por sua vez, segundo os comentários do código fonte pode representar uma falha na transmissão da informação ou representar que a conexão sem fios está desligada [37].

3.5.6 *Return Disable WIFI State*

Existem vários códigos que representam informação corrente do estado da rede sem fios, o estado pode ser ligado, desligado, acabado de desligar, acabado de ligar ou desconhecido. O utilizador ao selecionar esta opção o estado da rede sem fios será transmitido que está desligado, que equivale ao valor constante 1.

3.5.7 *Block Extra Info WIFI*

Ao selecionar esta opção não será transmitida informação adicional sobre a conexão de rede sem fios presente, após teste do método intitulado de "getExtraInfo" localizado em "android.net.NetworkInfo"a única informação que era transmitida foi somente o nome da rede, ou seja, o SSID de qualquer das formas o utilizador ao selecionar esta opção não será transmitida nenhuma informação extra, ou seja, é transmitido “null” que indica que nenhuma rede sem fios está disponível [35].

Nome	Método	Classe
MAC WLAN Inferface	getMacAddress	android.net.wifi.WifiInfo
SSID	getSSID	android.net.wifi.WifiInfo
BSSID	getBSSID	android.net.wifi.WifiInfo
Block list APs found	getScanResults	android.net.wifi.WifiManager
Block list of Configured Networks	getConfiguredNetworks	android.net.wifi.WifiManager
Return Disable WIFI State	getWifiState	android.net.wifi.WifiManager
Block Extra Info WIFI	getExtraInfo	android.net.NetworkInfo
MAC Bluetooth interface	getAddress	android.bluetooth.BluetoothAdapter

Tabela 3.3: Classes e métodos onde são efetuados os processos de *hook* no grupo *Network*.

3.5.8 *MAC Bluetooth Interface*

Tal como na subsecção 3.5.1 e 3.5.3 caso o utilizador selecione esta opção será gerado um endereço MAC, mas neste caso o endereço representa a interface de Bluethooth.

3.6 Grupo de restrição *Localization*

O Android fornece diversos tipos de acessos à localização do dispositivo, tais como, sensor de GPS, rede sem fios e através de acesso à rede móvel. Como podemos confirmar em [38] [8] os serviços de localização são dos mais utilizados e acedidos pelas aplicações.

De facto, existem aplicações que necessitam de aceder à localização dos seus utilizadores para realizarem as suas funcionalidades, mas qual a razão de transmitirem essa informação para servidores externos? Será para monitorização dos seus utilizadores? Se sim, qual será frequência de acesso a estes dados? Se o acesso e transmissão desses dados for efetuado diversas vezes ao dia, temos um sério problema de segurança como de privacidade, pois a localização revela costumes da vida diária, como os trajetos que efetuamos no nosso dia a dia, os locais onde permanecemos mais tempo. Informação que divulga em muito a nossa intimidade da vida privada e familiar.

Com a utilização do IdentitySpoofing o utilizador poderá selecionar um de oito países, onde estão incluídos Portugal, Espanha, França, Itália, Alemanha, Roménia, Índia, China e as suas informações de localização serão transmitidas consoante o país selecionado independentemente do tipo de acesso que esteja a utilizar para aceder à localização do dispositivo.

Existem métodos no Android que não estão diretamente interligados com a localização mas as aplicações ao acederem a essa informação e com cruzamento de dados conseguem obter informação da localização do utilizador embora não exatamente precisa. Toda a estratégia utilizada e todos os métodos que possam revelar informação do utilizador será abordada nos seguintes subsecção.

3.6.1 Estratégia utilizada

A plataforma Android permite aos seus programadores obterem informação de localização dos seus clientes através das diferentes tecnólogas que é o caso do GPS e dados móveis. Para cada uma destas tecnologias existem diversos métodos que fornecem a mesma informação, podendo esta informação estar formatada de forma diferente, ou não estar tão completa. Por exemplo, se o utilizador estiver a utilizar a tecnologia GPS as aplicações podem aceder ao valor da latitude através do método "getLatitude" que pode ser consultado em [34], no entanto o método "getLastKnowLocation" localizada na mesma classe fornece igualmente a coordenada da latitude e mais informação adicional, tal como, a longitude. Devido a estes pormenores foi necessária a criação da base de dados *usage* para que, consigamos guardar por um intervalo de tempo de 5 minutos a localização acedida por uma determinada aplicação quando o acesso é feito através do sensor de GPS ou 10 minutos se o acesso é feito através dos dados móveis. Após terminado o intervalo de tempo a localização será selecionada novamente aleatoriamente consoante o país escolhido pelo utilizador.

Na base de dados usage é inserida a informação do UID da aplicação. O método que corresponde ao código do país da rede móvel, de forma a conseguir identificar qual selecionada pelo utilizador. O nome do *package* da aplicação. O booleano *restricted*, que neste caso tem valor nulo devido a não ser necessário. O nome do grupo que estamos a restringir, neste caso, correspondente ao grupo "*Localization*". O tempo do último acesso em milissegundos. O número da linha da base de dados *fakedata* correspondente à última localização fornecida ao utilizador. E por final, no campo extra é inserido qual o tipo de tecnologia utilizada para o acesso da localização (GPS ou rede móvel). Só desta forma, é que conseguimos garantir que os dados que são transmitidos às aplicações são reais embora sejam pertencentes a outros países.

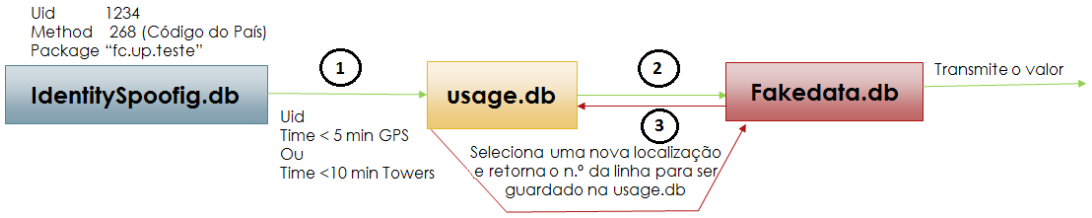


Figura 3.8: Modelo de funcionamento das bases de dados para a localização.

Na Figura 3.8 está representada um simples modelo de como interagem as três bases de dados. Inicialmente na etapa **número 1** o programa verifica se existe o UID da aplicação que efetuou o pedido de localização na base de dados **Usage**. Se existir, verifica se o acesso foi a menos de 5 minutos para um método que pertença ao sensor de GPS ou se foi a menos de 10 minutos, se pertencer a métodos relativos ao acesso através de dados móveis. Caso seja afirmativo, passamos para a etapa **número 2**, nesta etapa o programa já sabe qual a linha da tabela na base de dados **Fakedata** que tem de retribuir os valores e seguidamente transmite o valor. Caso não existisse o UID na etapa **número 1** ou se existir mas o tempo for maior que 5 ou 10 minutos consoante a tecnologia utilizada para aceder à localização. Primeiro o programa verifica qual o

país selecionado pelo utilizador, depois faz a pesquisa da localização na base de dados **Fakedata**, transmite essa informação e insere na base de dados **Usage** toda a informação mencionada anteriormente, o que corresponde à etapa **número 3**.

3.6.2 Acesso por GPS

Nesta subsecção vamos abordar os métodos que são utilizados para fornecer as coordenadas de GPS e a análise do processo de *hook* que efetuamos. Todos os processos de *hook* podem ser consultados na classe principal IdentitySpoofing.java e as classes e métodos de cada processo de *hook* pode ser consultado na Tabela 3.4.

1. Método **getLastKnownLocation**:

   O programador através deste método consegue aceder às últimas informações de localização acedidas pelo dispositivo, podendo o sensor de GPS ou as redes sem fios estarem desligadas. O retorno deste método é uma lista contendo informação do tipo de acesso, latitude, longitude, estimativa de precisão, tempo em milissegundos desde 1 Janeiro 1970, altitude, a velocidade se disponível, *bear* é o valor em graus que corresponde à direção horizontal do dispositivo e por último temos informação adicional especifica do provedor sobre a correção de localização, podendo este valor ser nulo. Um exemplo de resposta seria:

   ```
   Location [gps, 65, 966700,-18533300, acc=1 et=+37s655ms alt=15.0444 vel=0.0 bear=0.0, null]
   ```

   Se no menu *Localization* for selecionado um dos países disponíveis a aplicação irá realizar o processo de *hook*, os parâmetros serão alterados na classe "XLocationGpsManager.java", neste caso particular, os únicos dois campos modificados são o campo referente à latitude e longitude todos os outros se estiverem disponíveis ficaram iguais.

2. Método **requestLocationUpdates**

   Método que permite às aplicações receberem notificações com os novos dados de localização dos seus utilizadores. Este método tem quatro parâmetros: o nome do fornecedor com o qual se registar, o tempo mínimo entre as atualizações da localização, a mínima distância das atualizações de localização e um método que será chamado para cada atualização de localização.

   Como resposta serão transmitidos novos dados de latitude, longitude, tempo e a estimativa de precisão.

3. Métodos **getLatitude** e **getLongitude**

   O programador pode aceder somente às coordenadas de latitude e de longitude individualmente ao utilizar um destes métodos, respetivamente. Ao selecionar esta opção será transmitido coordenadas referente ao país selecionado pelo utilizador.

Método	Classe
getLatitude	android.location.Location
getLongitude	android.location.Location
getLastKnownLocation	android.location.LocationManager
requestLocationUpdates	android.location.LocationManager

Tabela 3.4: Classes e métodos onde são efetuados os processos de *hook* no grupo *Localization* através de GPS

3.6.3 Dados móveis

Relativamente às redes móveis existem três tecnologias dominantes que são GSM (Global System for Mobile Communications), CDMA (Code Division Multiple Access) e LTE (Long Term Evolution). Globalmente GSM é bastante mais utilizada como podemos verificar em [12] tornando-se o modelo padrão na Europa em 1980 [47], no entanto CDMA é bastante utilizada nos Estados Unidos e as redes LTE será a mais utilizada num prazo de 4 anos como podemos verificar em [12]. No presente trabalho abordaremos apenas a tecnologia GSM devido a ser mais utilizada atualmente e ao dispositivo utilizado não suportar as tecnologias LTE e CDMA.

Todos os processos de *hook* podem ser consultados na classe principal IdentitySpoofing.java e as classes e métodos de cada processo de *hook* pode ser consultado na Tabela 3.5.

1. Método **getMcc**:

   O MCC como dito anteriormente representa o código do país da rede móvel, se para uma determinada aplicação um dos oito países estiver selecionado, a componente de *software* irá consultar a base de dados usage para identificar se o acesso à localização foi realizado nos últimos dez minutos. Se sim, transmite o valor correspondente ao MCC da determinada linha da base de dados fakedata; se não, realiza uma nova pesquisa na base de dados fakedata na tabela "towers" e retorna o valor.

2. Método **getMnc**:

   O MNC representa o código da operadora do serviço móvel, em Portugal, por exemplo, a operadora Vodafone é representada pelo código 01 e a operadora MEO é representada pelo código 02. De forma análoga ao MCC, este valor será transmitido consoante o país selecionado pelo utilizador.

3. Método **getCid**:

   CID (Cell Identifier) representa o identificador único das antenas de acesso à rede móvel. Todo o processo de transmissão do valor é igual aos anteriores referentes à subsecção 3.6.3.

4. Método **getLac**:

   Um país é dividido em diferentes áreas o LAC (Location Area Code) é um código que identifica cada uma dessas áreas.Todo o processo de transmissão do valor é igual aos anteriores referentes à subsecção 3.6.3.

Método	Classe
getMcc	android.telephony.CellIdentityGsm
getMnc	android.telephony.CellIdentityGsm
getCid	android.telephony.CellIdentityGsm
getLac	android.telephony.CellIdentityGsm
getCellLocation	android.telephony.TelephonyManager
getAllCellInfo	android.telephony.TelephonyManager

Tabela 3.5: Classes e métodos onde são efetuados os processos de *hook* no grupo *Localization* através de dados móveis.

É importante salientar que as aplicações ao consultarem um único destes quatro métodos não têm informação suficiente da localização do dispositivo, mas ao realizarem pedidos a todos os métodos anteriores já é possível adquirir uma boa referência localização do dispositivo, mesmo não sendo 100% correta, a sua fiabilidade já é uma boa garantia.

5. Método **getCellLocation**:

   Ao utilizar o método "getCellLocation" fornece-nos acesso ao CID e LAC em um tuplo. Em análise ao processo de *hook* o primeiro parâmetro é referente à classe "android.telephony.TelephonyManager" [36] no entanto, os anteriores métodos são referentes à classe "android.telephony.CellIdentityGsm" [33] e ambos retornam a mesma informação. Este tipo de casos são bastante vulgares e é uma das etapas mais complicada na realização de uma aplicação desta natureza, pois podemos aceder à mesma informação de formas diferentes.

6. Método **getAllCellInfo**:

   Para a utilização deste método é necessária a permissão de ACCESS_COARSE_UPDATES, o seu propósito é transmitir todo o tipo de informação de célula disponíveis no dispositivo (MCC;CID;MNC;LAC). No entanto, o processo de *hook* deste método irá transmitir nenhuma informação, ou seja, “null” que representa que a informação não está disponível.

3.6.4 Acesso por GPS e dados móveis

1. Métodos **getLatitude** e **getLongitude**:

   Ambos os métodos estão localizados na classe "android.location.Geofence". Os serviços de *geofence* ou área demarcada permite combinar a localização do utilizador com a sua proximidade de um local de interesse especificando a latitude e longitude desse local. Pode ser utilizado por aplicações de retalho para cativarem clientes, como por exemplo, o envio-o de mensagem a informar que oferece 20% de desconto em qualquer produto ao mostrar a SMS na loja em um intervalo de tempo. O utilizador só receberá a mensagem se estiver num raio de alguns quilómetros. Ao selecionar um país serão transmitidas coordenas de latitude e longitude consoante o país selecionado.

Método	**Classe**
getNetworkOperator	android.telephony.TelephonyManager
getSimOperator	android.telephony.TelephonyManager
getSimCountryIso	android.telephony.TelephonyManager
getNetworkCountryIso	android.telephony.TelephonyManager
getSimOperatorName	android.telephony.TelephonyManager
getNetworkCountryName	android.telephony.TelephonyManager
getSimSerialNumber	android.telephony.TelephonyManager

Tabela 3.6: Classes e métodos onde são efetuados os processos de *hook* no grupo *Localization* em relação a informações do cartão SIM.

3.6.5 Informações do cartão SIM

A informação relativa aos cartões SIM dos dispositivo são baseadas no país que fornece o serviço, desta forma, decidimos incluir todos métodos neste grupo de restrições. Todos os processos de *hook* podem ser consultados na classe principal IdentitySpoofing.java e as classes e métodos de cada processo de *hook* pode ser consultado na Tabela 3.6.

1. Métodos **getNetworkOperator** e **getSimOperator**:

   Relativamente a estes dois métodos têm estruturas similares, no método *getNetworkOperator* é retornado pelo Android o MCC seguido do MNC e no método *getSimOperator* é retornado MCC seguido do número 0 e seguido do MNC. Ambos os valores são transmitidos consoante o país selecionado pelo utilizador.

2. Métodos **getSimCountryIso** e **getNetworkCountryIso**:

   Ambos os métodos retornam a norma ISO 3166 que define os códigos dos nomes dos países, por exemplo, no caso de Portugal este método irá retornar a sigla "pt". Ao realizar o processo de *hook* será retornado as siglas de um dos 8 países disponíveis.

3. Métodos **getSimOperatorName** e **getNetworkCountryName**:

   Através de ambos os métodos é possível aceder ao nome do fornecedor do cartão SIM. Quando um país está selecionado pelo utilizador será transmitida o nome de uma organização que fornecem estes serviços no determinado país. No entanto, só temos dois nomes de organizações por cada país.

   Este tipo de informação é proveniente da classe TelephonyManager.java e não necessita de permissões para ser acedida. Logo as aplicações através da informação destes métodos, podem efetuar anúncios direcionados para tentarem convencer os utilizadores, por exemplo, a mudar de operadora oferecendo serviços especiais [6].

4. Método **getSimSerialNumber**:

   Representa o número de série do cartão SIM, também designado por ICCID (Integrated Circuit Card ID) a sua estrutura pode ser consultado na Figura 3.9.

O campo MII (Major Industry Identifier) é representada pela constante 89 que se refere à Administração de Telecomunicações e Agências de Operações Privadas. O campo Indicativo é referente ao indicativo do país, no caso de Portugal, corresponde ao valor 351. O MNC representa código da rede móvel, os restantes campos são identificadores utilizados pelo fornecedor.

Quando o utilizador seleciona um pais todos os campos são alterados inteligentemente consoante o país selecionado, os únicos números gerados são os dígitos que representam o fornecedor, a data e os restantes 6 dígitos.

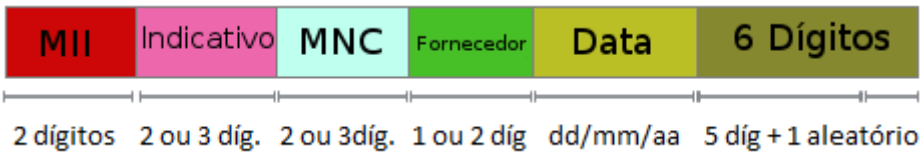


Figura 3.9: Estrutura do número de série do cartão SIM

5. Método **getSubscriberId**:

   O IMSI (International Mobile Subscriber Identity) é representado por um número único de 13 a 15 dígitos associados a todos os utilizadores de telemóveis GSM e UMTS (Universal Mobile Telecommunication System). Os primeiros 3 dígitos representam MCC, os seguintes 2 ou 3 dígitos representam o MNC e os restantes dígitos representam MSIN que serve para operadora identificar o dispositivo móvel.

   Ao ativar esta restrição o IdentitySpoofing consulta a base de dados fakedata seleciona aleatoriamente uma linha da tabela correspondente ao país selecionado e retira os valores do MCC e MNC. Seguidamente é gerado um número de 9 dígitos para concatenar aos valores MCC e MNC, a composição do IMSI está representada na Figura 3.10.

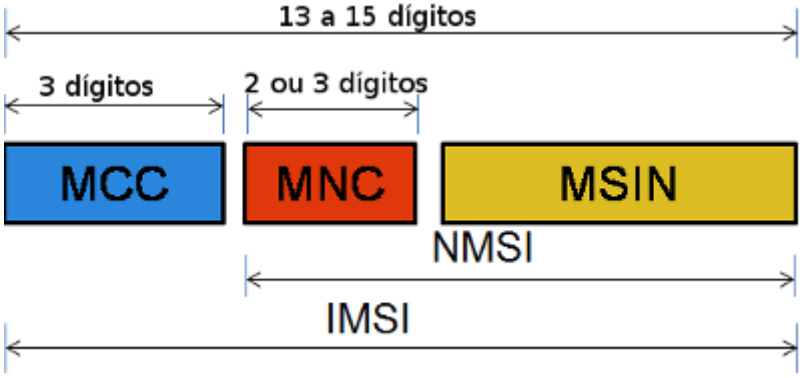


Figura 3.10: Estrutura do IMSI

3.7 Grupo de restrição *Media*

O utilizador no grupo de restrição *Media* pode restringir o acesso das aplicações aos ficheiros de imagens, vídeo e áudio quer estejam guardados em memória interna ou externa, para isso foi necessário intersetar os pedidos que são efetuados ao Fornecedor de Conteúdos por parte das aplicações e seguidamente simular que não existem nenhum registo de imagens, vídeo ou de áudio. Os URI's intersetados são:

1. "content://media/internal/images", para ficheiros de imagens guardadas em memoria interna.
2. "content://media/internal/video", para ficheiros de vídeos guardadas em memoria interna.
3. "content://media/internal/audio", para ficheiros de áudio guardadas em memoria interna.
4. "content://media/external/images", para ficheiros de imagens guardadas em memoria externa.
5. "content://media/external/video", para ficheiros de vídeos guardadas em memoria externa
6. "content://media/external/audio", para ficheiros de áudio guardadas em memoria externa

3.8 *Information Accessed By Google*

A Google dispõe de um Painel de Controlo [31] onde os seus utilizadores podem aceder informações suas coletadas pela Google. Muitas das vezes, o problema está em o utilizador não saber que está a fornecer tais dados, devido a não ler o seu contrato Google quando criou a conta, ou não definir com atenção as políticas de privacidade da sua conta [28].

Através do painel de controlo da Google o utilizador consegue obter informação sobre:

- A sua conta Google
- Os seus Dispositivos
- Os seus Contactos de telefone
- Histórico de Localizações
- Históricos de pesquisas e marcadores
- Entre outras

Para aceder ao menu *Settings* do IdentitySpoofing o utilizador necessita de ir às definições da aplicação através do botão "menu" do Android ou se o dispositivo não conter o botão é possível aceder através do botão disponível canto superior direito da Atividade principal.

É importante referir que cada alteração efetuada neste menu será necessário reiniciar o dispositivo para que este inicie com as novas configurações por defeito.

Caso o utilizador selecionar a opção "*Random Country Location*" e seguidamente, reiniciar o dispositivo. Por cada pedido efetuado relativamente à localização por parte de alguma aplicação ou serviços da Google será sempre transmitida um novo valor secionado aleatoriamente da base de dados *Fakedata*. A classe que efetuamos o processo de *hook* é "android.location.LocationManager.ListenerTransport" e o método é "onLocationChanged".

Todo o processo de *hook* pode ser analisado na classe principal IdentitySpoofing. Para este serviço funcionar corretamente o dispositivo deverá ter sinal de GPS e os seus dados móveis ativos.

O utilizador através deste menu pode ainda restringir o acesso aos contatos, histórico e marcadores do navegador de Internet e informação do calendário. Este cenário é idêntico aos processos efetuados para o grupo *Media* 3.7.

Este serviço só foi implementado no IdentitySpoofing em todos os sistemas estudados no capítulo anterior nenhum efetua este tipo de restrição.

Capítulo 4

Testes e Resultados

Para a realização dos testes foi desenvolvida uma nova aplicação que abusa das permissões disponíveis para o sistema Android, de forma a, conseguirmos testar os métodos que restringimos com a aplicação IdentitySpoogfing.

A aplicação é composta somente por uma Atividade principal contendo apenas um botão para inicializar a pesquisa da informação proveniente no dispositivo.

Após a pesquisa são criados 17 ficheiros JSON (JavaScript Object Notation) e seguidamente transferimos todos os ficheiros para um servidor para realizarmos o tratamento da informação.

O teste foi efetuados no Samsung GT-S7580 com a versão 4.2.2 do Android. No entanto, devido a versão (Lollipop) ser atualmente a mais utilizada criamos um dispositivo virtual com a versão 5.1 no *software* Genymotion [23] Seguidamente, com o apoio do tutorial [1] realizarmos o *flash* ao dispositivo e instalamos a *framework* Xposed. Todos os resultados são iguais aos apresentados, no entanto o endereço físico do Bluetooth e o número de telefone não são acedidos: com ou sem ação do IdentitySpoofing.

Nas seguintes subseções iremos analisar os dados transmitidos com e sem intervenção do IdentitySpoofing para cada um dos grupos de restrições.

4.1 Grupo *Advertising Identification*

Na Figura 4.1 estão representados os identificadores de publicidade sem intervenção do IdentitySpoofing e na Figura 4.2 está representada os valores com a intervenção do IdentitySpoofing. Como podemos verificar em ambos os métodos conseguimos transmitir valores completamente diferentes do original.

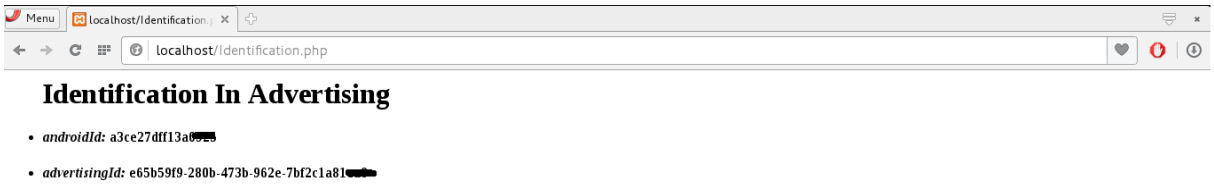


Figura 4.1: Identificadores de publicidade legítimos.

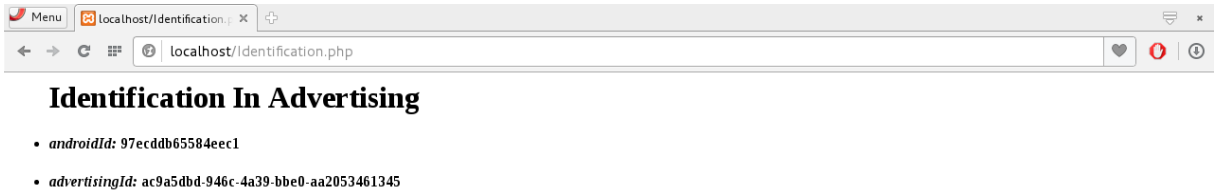


Figura 4.2: Identificadores de publicidade com a componente de *software*.

Figura 4.3: Menu *Phone* dados legítimos.

Figura 4.4: Menu *Phone* restringido pela a componente de *software*.

4.2 Grupo *Phone*

Na Figura 4.3 está representado o número de série, o número de telefone e o IMEI do dispositivo. Todos estes campos não foram restringidos pelo IdentitySpoofing, em contrapartida a Figura 4.4 reprenta os mesmos campos mas restringidos pela componente de *software*. É importante referir que no acesso legítimo ao número de telefone não é apresentado devido a propriedades do dispositivo, no entanto, quando efetuamos o processo de *hook* o número de telefone é gerado e apresentado.

Na Tabela 4.1 e Figura 4.2 estão representadas dois excertos das aplicações instaladas no dispositivo e processos que estão em execução, respetivamente. Ao ativarmos o IdentitySpoofing para estas duas restrições nenhuma informação será transmitida. Como podemos verificar na Tabela 4.3 e 4.4.

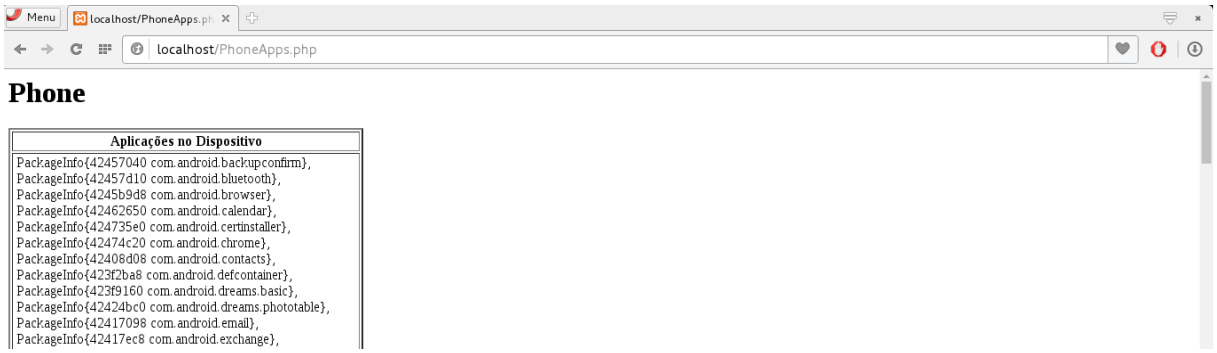


Tabela 4.1: Excerto da tabela que representa as aplicações instaladas no dispositivo.

Tabela 4.2: Excerto da tabela que representa os processos em execução no dispositivo.

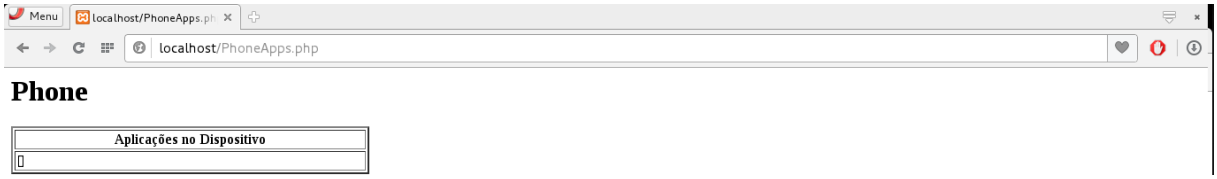


Tabela 4.3: Tabela que representa as aplicações instaladas no dispositivo com IdentitySpoofing ativo.

Tabela 4.4: Tabela que representa os processos em execução no dispositivo com IdentitySpoofing ativo.

As restantes informações do grupo *Phone* são acedíveis através do fornecedor de conteúdos do Android. Na Figura 4.5 está representada um excerto da lista de contactos com a informação do identificador único de cada contacto, nome e o número de telefone. Podiam ser adicionados mais campos à tabela, como por exemplo, o endereço de localização da imagem do contacto, endereço eletrónico, morada, contacto alternativo, entre outros. Na Figura 4.6 representada a informação que obtemos quando o IdentitySpoofing está ativo, ou seja, nenhuma informação de contactos é transmitida à aplicação.

Na Figura 4.7 estão representados informações sobre o calendário do dispositivo nem todos os campos possíveis foram selecionados e transferidos para o servidor. Podemos verificar os diferentes

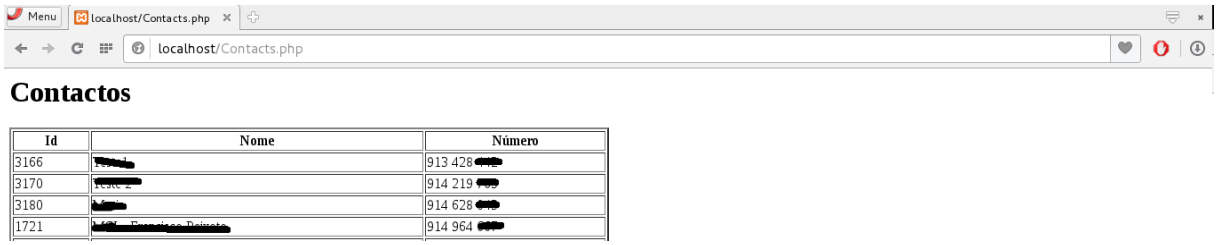


Figura 4.5: Menu *Phone* dados legítimos dos contactos.

Figura 4.6: Menu *Phone* com contactos restringidos pela a componente de *software*.

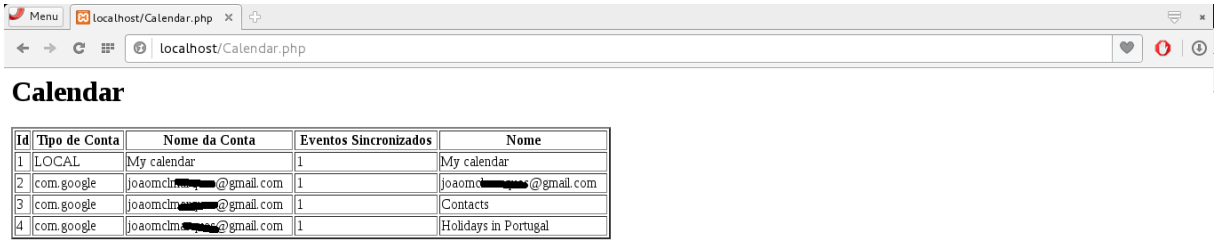


Figura 4.7: Menu *Phone* dados legítimos dos calendários.

Figura 4.8: Menu *Phone* com acesso aos calendário restringido pela a componente de *software*.

tipos de calendários disponíveis no dispositivo, assim como, as contas de correio electrónico a que estão associadas ao calendário e o número de eventos sincronizados de cada conta. Na Figura 4.8 nenhuma informação é transmitida devido a estar restringida pelo IdentitySpoofing.

Na Figura 4.9 está representada os registos de chamadas com a informação do nome, número do contacto, data e duração sem intervenção do IdentitySpoofing. No entanto, na Figura 4.10 nenhuma informação é exibida devido ao IdentitySpoofing proteger igualmente este tipo de acesso a informação sensível do utilizador.

Na Figura 4.11 está representada uma tabela com os registos do navegador de internet sem intervenção do IdentitySpoofing enquanto na Figura 4.12 a informação fornecida quando restringimos o acesso.

Na Figura 4.13 podemos verificar informação relativa a mensagens de texto e mensagens

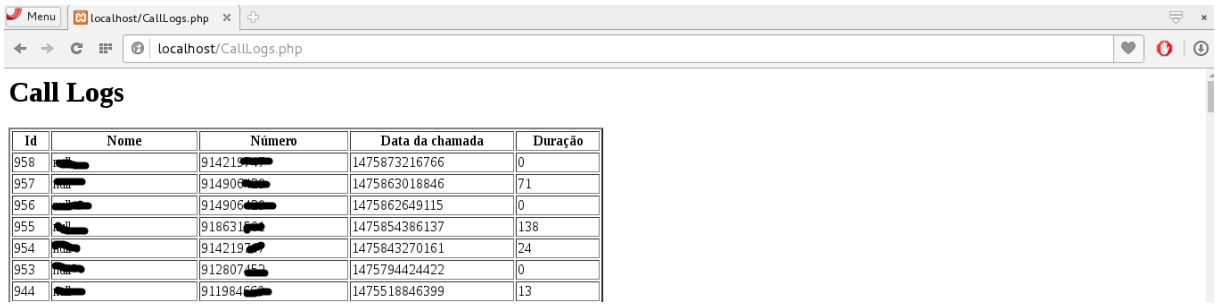


Figura 4.9: Menu *Phone* dados legítimos do registo de chamadas telefónicas.

Figura 4.10: Menu *Phone* com acesso ao registo de chamadas restringido pela a componente de *software*.

multimédia sem intervenção do IdentitySpoofing. No entanto na Figura 4.14 podemos verificar a mesma informação protegida pela componente de *software*.

4.3 Grupo *Localization*

Iremos abordar de seguida os testes realizados para o menu referente à localização do dispositivo, como referido anteriormente a localização pode ser obtida através do GPS e redes sem fios. Na Figura 4.15 está representado o acesso legítimo à localização do dispositivo através do GPS. Na Figura 4.16 está representado a localização graficamente utilizando o Google Mapas, como podemos analisar as coordenadas legítimas são em Portugal e é representado pelo indicador de cor vermelha.

Ao utilizarmos o IdentitySpoofing podemos selecionar a localização que queremos transmitir para cada uma das aplicações instaladas no dispositivo, para a aplicação de testes o país selecionado foi a Roménia e obtivemos os resultados que constam na Figura 4.17 e a sua representação gráfica está representada na Figura 4.18.

Se o acesso à localização for efetuado através dos dados móveis do dispositivo obtivemos as informações representadas na Figura 4.19 e na Figura 4.20 está a sua representação gráfica utilizando o Opencellid [21]. Com a utilização da componente de *software* novamente com o país Roménia selecionado obtemos as informações representadas na Figura 4.21 e a sua representação gráfica na Figura 4.22.

Existem identificadores do cartão SIM do dispositivo que são baseados igualmente na localização o acesso a essa informação está representada na Figura 4.23, no entanto quando

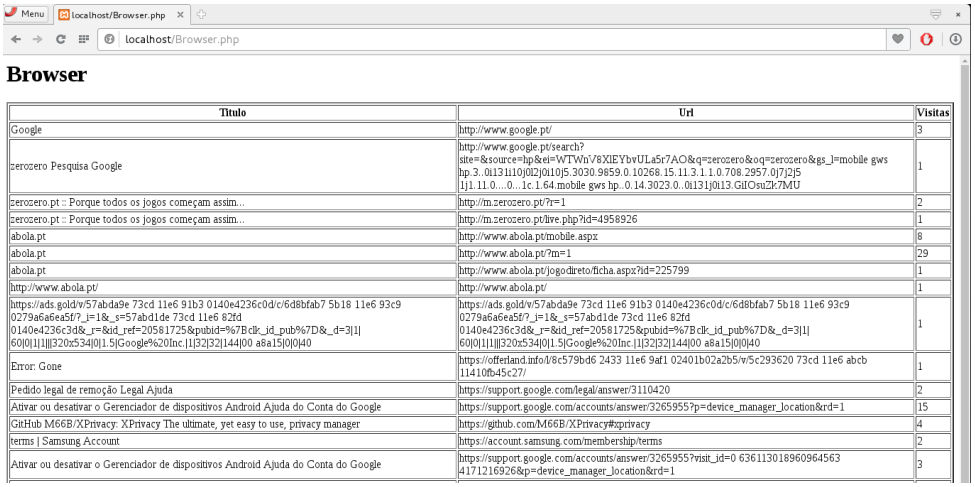


Figura 4.11: Menu *Phone* dados legítimos do acesso a dados do navegador de Internet.

Figura 4.12: Menu *Phone* com acesso a informações do navegador de Internet restringido pela a componente de *software*.

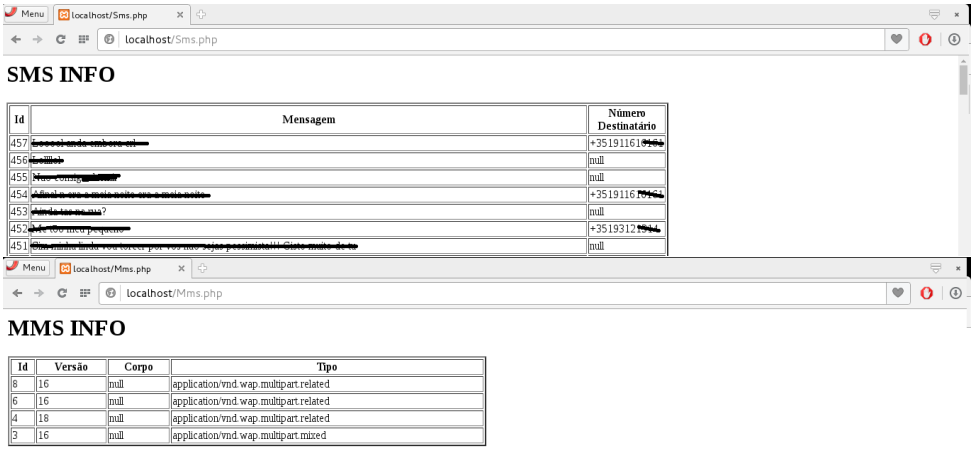


Figura 4.13: Acesso legítimo a mensagens de texto e de multimédia.

Figura 4.14: Acesso às mensagens de texto e de multimédia restringido pelo IdentitySpoofing.

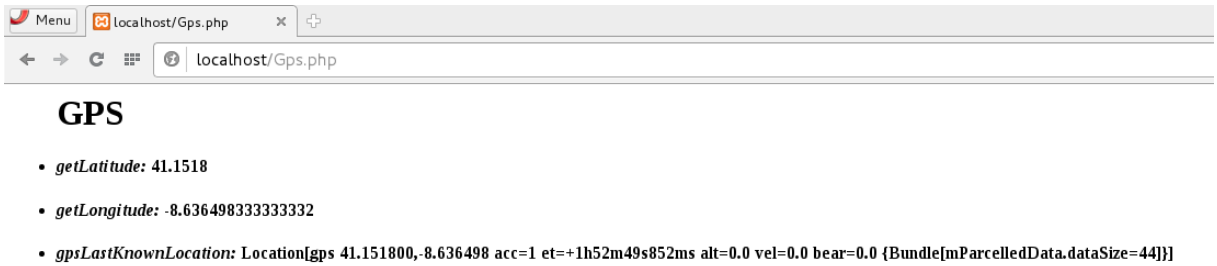


Figura 4.15: Dados legítimos do GPS.

Figura 4.16: Mapa retirado do Google Mapas com os dados legítimos.

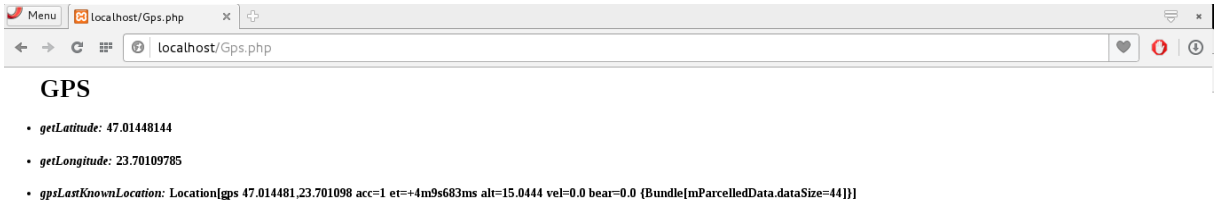


Figura 4.17: Dados do GPS com intervenção do IdentitySpoofing selecionado o país Roménia.

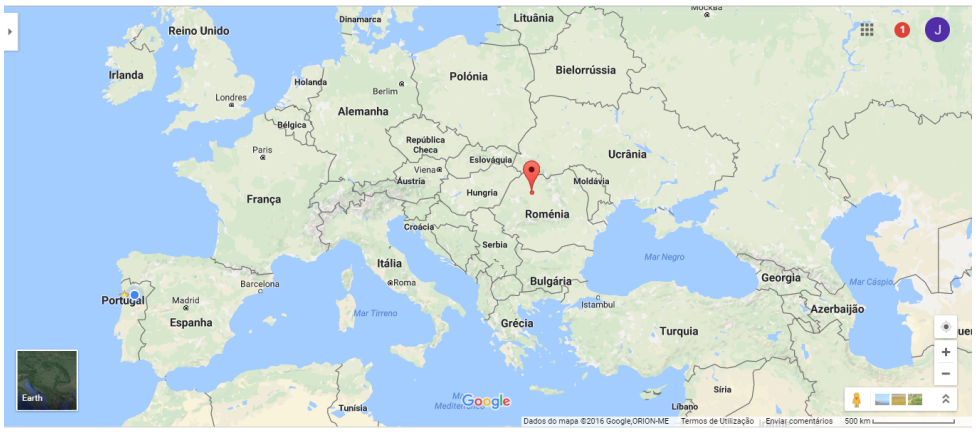


Figura 4.18: Mapa retirado do Google Mapas com os dados da intervenção do IdentitySpoofing.

Figura 4.19: Dados legítimos de acesso à localização através de dados móveis.

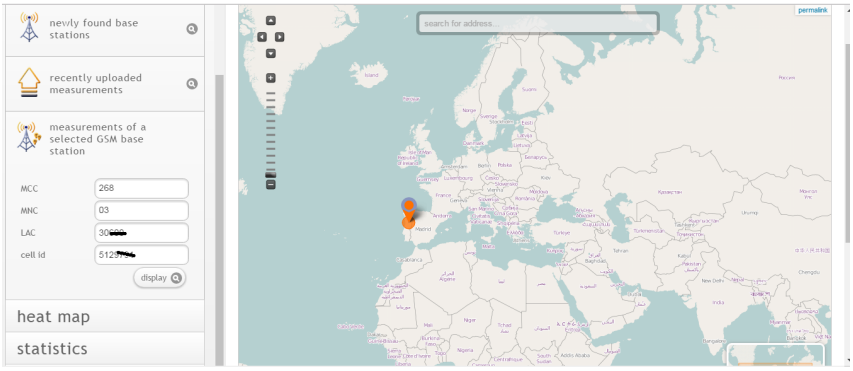


Figura 4.20: Representação gráfica dos dados legítimos.

Figura 4.21: Dados de localização através de dados móveis após escolha do país Roménia para a aplicação de Testes.

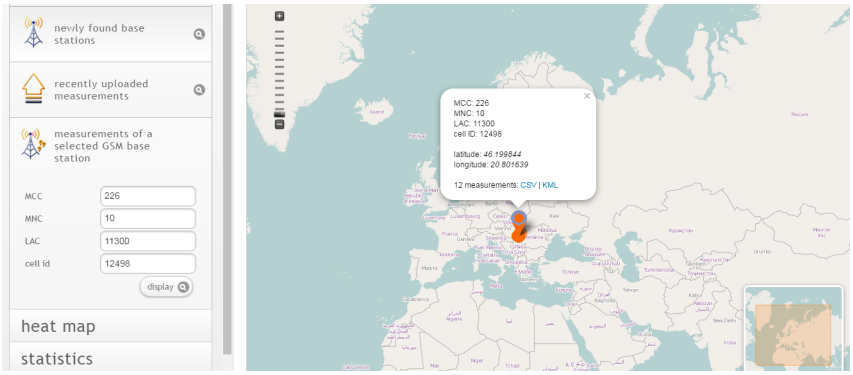


Figura 4.22: Representação gráfica dos dados obtidos cojm o IdentitySpoofing ativo.

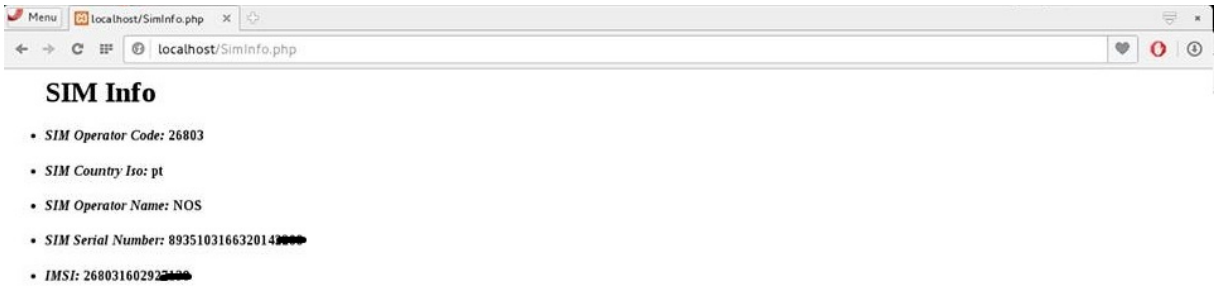


Figura 4.23: Dados legítimos do acesso a informações do cartão SIM.

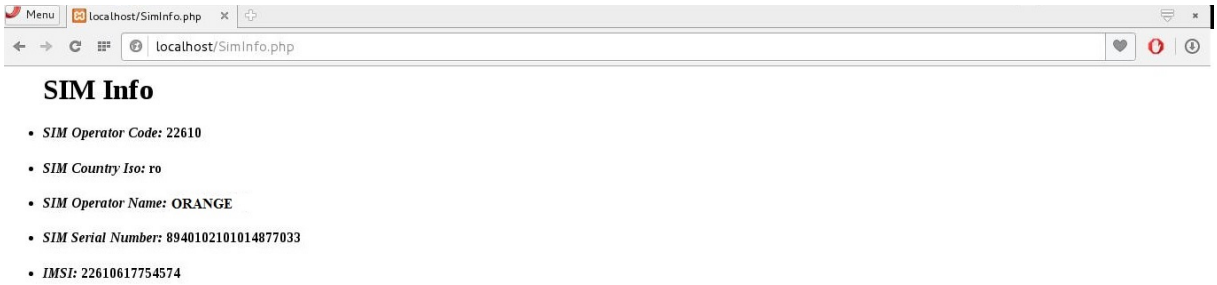


Figura 4.24: Dados do cartão SIM quando selecionado o país Roménia para a aplicação de Testes.

pretendemos transmitir informação de localização de outro país, neste caso de exemplo a Roménia a informação transmitida está representada na Figura 4.24.

4.4 Grupo *Network*

A informação com acesso legítimo a informação de redes sem fios do dispositivo, tais como, nome da rede, endereço MAC da placa de rede sem fios, do *router* e da interface de Bluetooth, estado de conexão da rede, lista de configurações de redes onde podemos encontrar diversas informações desde os endereços IP's dos routers, protocolos permitidos, o canal onde opera, entre outras, está representada na Figura 4.25. Na Figura 4.5 temos a informação sobre os pontos de acesso possíveis para conexão. Com a utilização do IdentitySpoofing é possível transmitir informação gerada para todos os campos, excluindo a lista de configuração das redes sem fios e das informações dos pontos de acesso, onde somente restringimos o acesso às aplicações, como podemos verificar na Figura 4.26.

4.5 Grupo *Média*

Relativamente aos conteúdos de imagens, vídeos e áudio podem ser consultados na Figura 4.27, 4.28, 4.29, respetivamente. Ao restringirmos as aplicações a estes tipos de acessos obtemos os resultados apresentados nas Figuras 4.30, 4.31, 4.32.

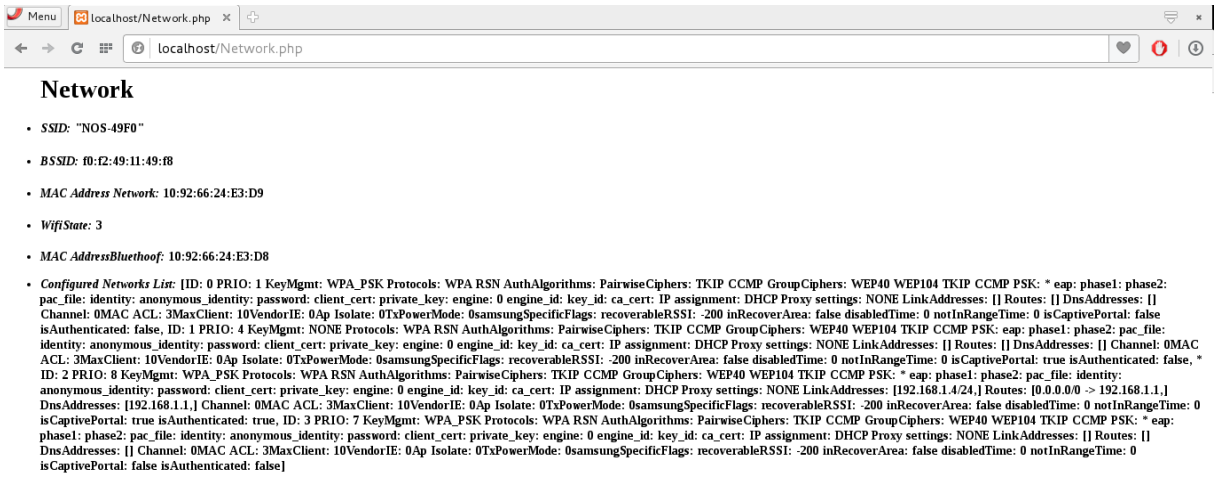


Figura 4.25: Dados legítimos do acesso a informações de rede.

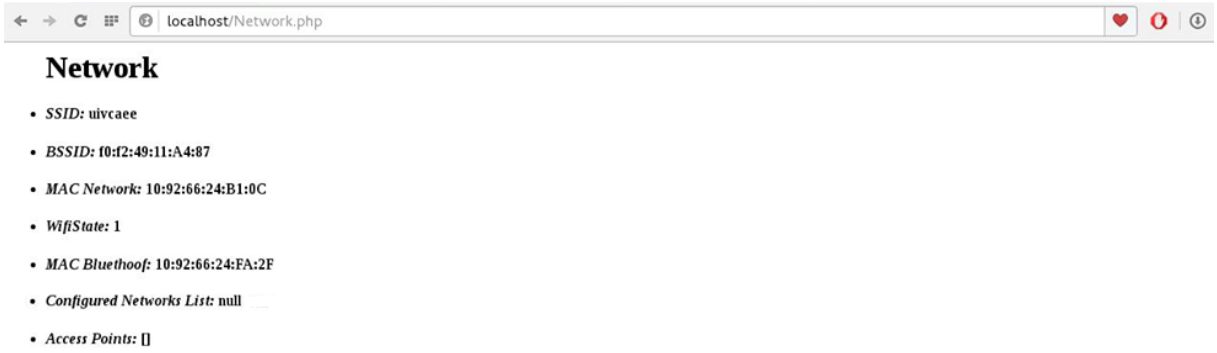


Figura 4.26: Dados restringidos pela componente de *software*.

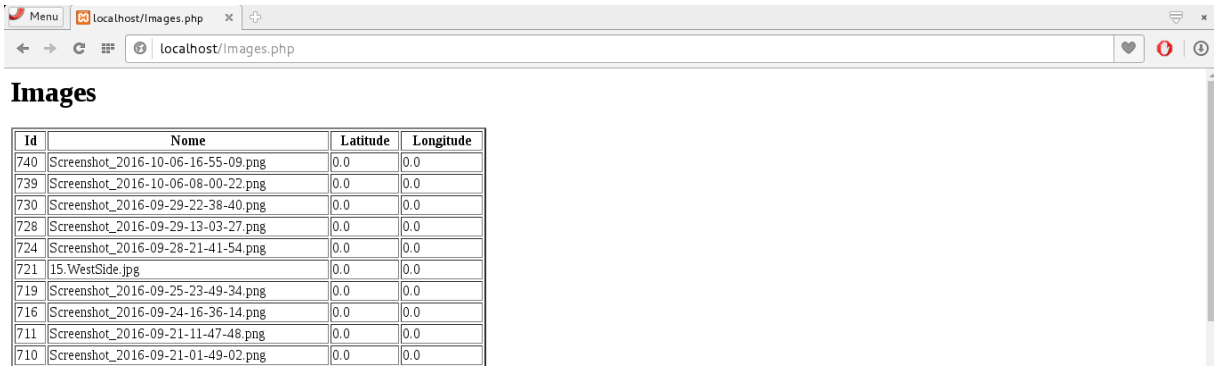
Capacidades	Nível	Frequeência	Carimbo de tempo
capabilities: [WPA PSK TKIP+CCMP][WPA2 PSK TKIP+CCMP][WPS][ESS]	level: 55	frequency: 2427	timestamp: 102141090305
capabilities: [WPA PSK TKIP+CCMP][WPA2 PSK TKIP+CCMP][WPS][ESS]	level: 58	frequency: 2447	timestamp: 102141090549
capabilities: [WPA PSK TKIP+CCMP][WPA2 PSK TKIP+CCMP][ESS]	level: 71	frequency: 2417	timestamp: 102141090152
capabilities: [WPA PSK TKIP+CCMP][WPA2 PSK TKIP+CCMP][WPS][ESS]	level: 78	frequency: 2442	timestamp: 102141090396
capabilities: [WPA PSK TKIP+CCMP][WPA2 PSK TKIP+CCMP][WPS][ESS]	level: 76	frequency: 2472	timestamp: 102141090854
capabilities: [ESS]	level: 55	frequency: 2427	timestamp: 102141090244

Tabela 4.5: Tabela com informação sobre os pontos de acesso de redes móveis.

4.6 Informação rastreada pela Google

Para demonstrar a informação que conseguimos restringir à Google como contactos e localização socorremos ao painel de controlo da Google [31].

É importante realçar que todas as configurações efetuadas neste menu o dispositivo necessita de ser reiniciado.



Id	Nome	Latitude	Longitude
740	Screenshot_2016-10-06-16-55-09.png	0.0	0.0
739	Screenshot_2016-10-06-08-00-22.png	0.0	0.0
730	Screenshot_2016-09-29-22-38-40.png	0.0	0.0
728	Screenshot_2016-09-29-13-03-27.png	0.0	0.0
724	Screenshot_2016-09-28-21-41-54.png	0.0	0.0
721	15.WestSide.jpg	0.0	0.0
719	Screenshot_2016-09-25-23-49-34.png	0.0	0.0
716	Screenshot_2016-09-24-16-36-14.png	0.0	0.0
711	Screenshot_2016-09-21-11-47-48.png	0.0	0.0
710	Screenshot_2016-09-21-01-49-02.png	0.0	0.0

Figura 4.27: Acesso às imagens do dispositivo.

Videos

Id	Nome	Duração
703	20160919_191857.mp4	59049
702	20160919_191714.mp4	55167
700	20160919_190241.mp4	21525
699	20160919_184748.mp4	14122
697	20160919_184726.mp4	11541
696	20160919_184600.mp4	46441

Figura 4.28: Acesso às vídeos do dispositivo.

Audio

Id	Nome	Data	Duração
12	hangouts_video_call.ogg	1469213340	0
11	hangouts_message.ogg	1469213339	0
21	Over_the_horizon.mp3	1469213351	178964
5	Morning_flower.ogg	1469213345	30984
7	S_Whistle.ogg	1469213346	1620
9	S_Over_the_horizon.ogg	1469213346	50198
16	Dawn_chorus.ogg	1469213349	28282
17	Day_by_day.ogg	1469213349	35505
18	Sea_breeze.ogg	1469213349	30836
19	Serene_morning.ogg	1469213350	22750
20	Springtime_memories.ogg	1469213350	26796

Figura 4.29: Acesso a ficheiros de áudio do dispositivo.

Ao reiniciar, o dispositivo irá iniciar com as novas configurações, se alguma das aplicações pedir acesso, por exemplo, ao calendário o acesso será restringido por defeito a todas as aplicações, mesmo que a opção de bloquear o acesso ao calendário esteja desativo nas aplicações. Devido a esta configuração é necessário alguma precaução porque podemos necessitar do funcionamento correto de uma aplicação num dado instante. Para o utilizador modificar estas alterações terá sempre que reiniciar o dispositivo.

Na Figura 4.33 está representada a localização do dispositivo na China, como referido anteriormente, por cada pedido efetuado será feita uma nova pesquisa à base de dados e será retribuído esse valor.

Figura 4.30: Acesso a ficheiros de imagens restringidos pela componente de *software*.

Figura 4.31: Acesso a ficheiros de vídeos restringidos pela componente de *software*.

Figura 4.32: Acesso a ficheiros de áudio restringidos pela componente de *software*.

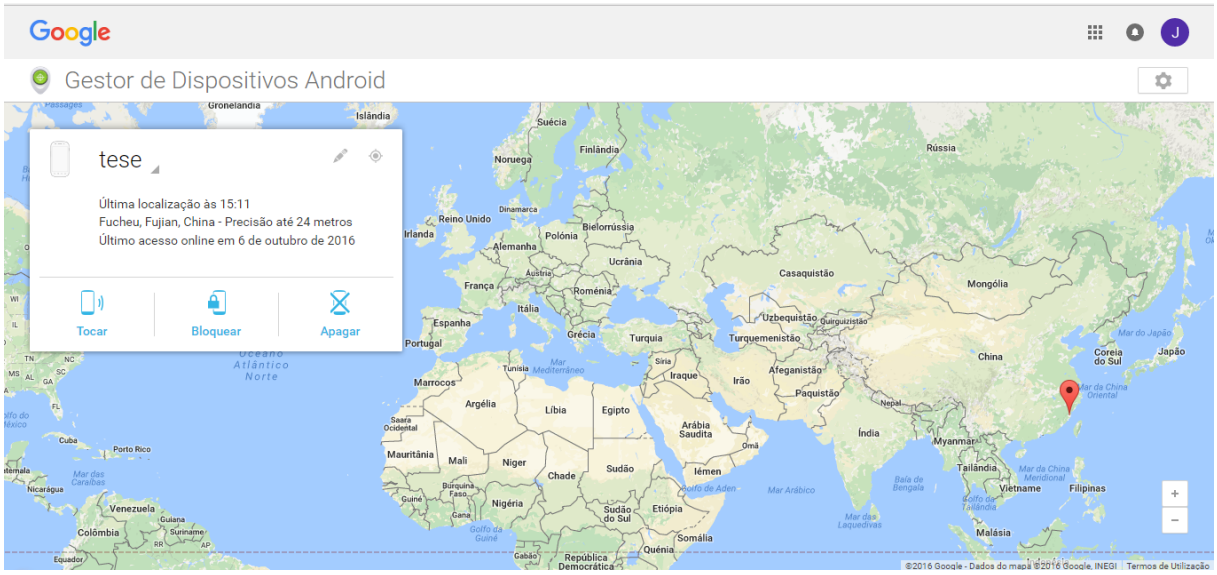


Figura 4.33: Localização do dispositivo na China.

4.7 Análise do comportamento do Xprivacy e DonkeyGuard com a aplicação de Testes

Com base na aplicação de testes desenvolvida analisaremos seguidamente os comportamentos das aplicações Xprivacy e DonkeyGuard. Recordo que ambas as aplicações também necessitam da *framework* Xposed para o seu funcionamento.

Como podemos verificar na Tabela 4.6 a primeira coluna é composta pelos grupos de permissões do IdentitySpoofing. Ao analisar o grupo *Advertising Identification* somente a

aplicação Xprivacy é que transmite os mesmos identificadores para todas as aplicações. O utilizador caso pretenda modificar um desses valores necessita de ir ao menu das definições e gerar um novo ou inserir manualmente. Tanto as aplicações DonkeyGuard e IdentitySpoofing transmitem diferentes identificadores de publicidade para cada uma das aplicações instaladas, no entanto o IdentitySpoonfing por cada pedido recebido gera sempre um novo valor.

Relativamente ao grupo *Phone* a aplicação Xprivacy transmite o mesmo número de série do dispositivo para todas as aplicações e este sem significado, apenas limita-se a modificar os dígitos que o compõem. DonkeyGuard gera diferentes valores para cada uma das aplicações instaladas mas igualmente sem significado, permitindo ainda que as aplicações acedam à informação das aplicações que estão instaladas e os processos que estão em execução.

Tanto o número de telefone como o IMEI são gerados por ambas as aplicações sem significado embora no Xprivacy esses valores continuem a ser o mesmo para todas as aplicações no dispositivo.

Com o IdentitySpoofing a geração dos valores é feita de forma inteligente para os identificadores terem significados e por cada pedido efetuado pelas aplicações será sempre gerado novos valores. No entanto, tem igualmente limitações a gerar o número de telefone pois limita-se a iniciar o número por 00 e seguidamente gera um conjunto de dígitos. O IMEI é gerado tendo em base o teste de Luhn.

Em análise ao grupo *Localization* o IdentitySpoofing destaca-se das aplicações em estudo devido a ter uma base de dados extra com informações de localização de 8 países diferentes, o que por sua vez se traduz na capacidade de transmitir informação dinâmica e com dados com significado para cada uma das aplicações instaladas no dispositivo mesmo que o acesso à localização seja feita através dos dados móveis.

Ao compararmos DonkeyGuard com Xprivacy nota-se que o Donkeyguard é mais limitado devido a não efetuar as devidas restrições de acesso a propriedades de localização do telemóvel quando está ligado através dos dados móveis. Não permite também a inserção ou negação de acesso a informações da operadora que fornece acesso de rede.

No entanto, o Xprivacy não gera os dados de uma forma inteligente, somente gera dados sem significado, como é o caso de todos os campos referente à localização do dispositivo quando acedida através dos dados móveis e igualmente dos dados da operadora responsável pelo cartão SIM, como por exemplo, os campos código da operadora, nome da operadora e o ISO do país em que operadora está a fornecer o serviço. Mas todos estes campos podem ser introduzidos pelo utilizador manualmente, embora o utilizador necessitará de ter conhecimento adequado para inserir dados que tenham significado.

Se o utilizador pretender criar um perfil mais real terá de modificar constantemente estes valores, pois a mobilidade faz parte dos telemóveis e estar sempre com as mesmas coordenadas do sensor do GPS e com os mesmos valores de localização referentes aos dados móveis não é um comportamento normal pois estamos sempre na mesma localização por longos períodos.

No grupo de restrições *Network* existe novamente uma limitação na aplicação DonkeyGuard pois não permite restringir o acesso à lista de configurações das redes sem fios e a informações dos pontos de acesso. Relativamente ao Xprivacy e ao IdentitySpoofing ambos protegem o acesso a estas duas características, todos os outros campos podem ser gerados pelo Xprivacy automaticamente ou o utilizador pode definir manualmente, mas será igualmente transmitido os mesmos valores para todas as aplicações e claro com valores sem significado, como é o caso dos endereços MAC.

No IdentitySpoofing todos os restantes dados são gerados para cada uma das aplicações e com mais inteligência.

Os conteúdos de média deveriam ser bloqueados nas três aplicações mas isso não aconteceu na aplicação Xprivacy, embora tenha a opção para restringir acesso aos ficheiros multimédia.

Relativamente a ser possível "enganar" os serviços da Google, como por exemplo, o gestor de localização fornecido pela Google somente o IdentitySpoofing o consegue fazer sendo necessário reiniciar o dispositivo para que inicie com as configurações origem modificadas.

Tabela 4.6: Tabela de comparação das aplicações em relação aos testes efetuados para o IdentitySpoofing.

	Xprivacy	**DonkeyGuard**	**IdentititySpoofing**
Advertising Identification			
Android Id	Sim mas o mesmo valor para todas as aplicações	Sim e dinâmico	Sim e dinâmico
Advertising Id	Sim mas o mesmo valor para todas as aplicações	Sim e dinâmico	Sim e dinâmico
Phone			
Número de Série Dispositivo	Sim mas o mesmo valor para todas as aplicações	Sim e dinâmico	Sim e dinâmico
Número de Telefone	Sim mas o mesmo valor para todas as aplicações	Sim e dinâmico	Sim e dinâmico
IMEI	Sim mas o mesmo valor para todas as aplicações	Sim e dinâmico	Sim e dinâmico
Processos em Execução	Sim bloqueia o acesso	Não	Sim bloqueia o acesso
Aplicações Instaladas	Sim bloqueia o acesso	Não	Sim bloqueia o acesso
Acesso aos Contactos	Sim bloqueia o acesso	Sim bloqueia o acesso	Sim bloqueia o acesso
Calendário	Sim bloqueia o acesso	Sim bloqueia o acesso	Sim bloqueia o acesso

Registo de chamadas	Sim bloqueia o acesso	Sim bloqueia o acesso	Sim bloqueia o acesso
Navegador Histórico e Marcadores	Sim bloqueia o acesso	Sim bloqueia o acesso	Sim bloqueia o acesso
SMS	Sim bloqueia o acesso	Sim bloqueia o acesso	Sim bloqueia o acesso
MMS	Sim bloqueia o acesso	Sim bloqueia o acesso	Sim bloqueia o acesso
Location			
Latitude GPS	Sim mas o mesmo valor para todas as aplicações	Sim mas valor constante sem intervenção do utilizador	Sim dinamicamente consoante o pais selecionado pelo utilizador
Longitude GPS	Sim mas o mesmo valor para todas as aplicações	Sim mas valor constante sem intervenção do utilizador	
Última localização conhecida GPS	Sim	Sim	
Nome da Operadora da Rede Móvel	Sim mas fornecido pelo utilizador e o mesmo para todas as aplicações	Não	
ISO do País fornecedor da Rede Móvel	Sim mas fornecido pelo utilizador e o mesmo para todas as aplicações	Não	
Código da Operadora de Rede Móvel	Sim mas mesmo valor para todas as aplicações (DEFACED)	Não	
Código do País da Rede Móvel (MCC)	Sim mas mesmo valor para todas as aplicações (001)	Não	
Código da Rede Móvel (MNC)	Sim mas mesmo valor para todas as aplicações (01)	Não	
Código da Area de Localização (LAC)	Sim mas fornecido pelo utilizador e o mesmo para todas as aplicações	Não	
Identificador Único da Célula (CID)	Sim mas fornecido pelo utilizador e o mesmo para todas as aplicações	Não	
Localização da célula GSM	Sim mas fornecido pelo utilizador e o mesmo para todas as aplicações	Não	
Última Localização Conhecida Rede	Sim	Sim	
Informação da célula (Cell Info)	Sim	Sim	
Código da Operadora do cartão SIM	Sim mas fornecido pelo utilizador e o mesmo para todas as aplicações	Não	
ISO do País do Cartão SIM	Sim mas mesmo valor para todas as aplicações (DEFACED)	Não	
Nome da Operadora do Cartão SIM	Sim mas mesmo valor para todas as aplicações (BA)	Não	
Número de Série do Cartão SIM	Sim mas mesmo valor para todas as aplicações	Sim	
Identificador de Subscrição (IMSI)	Sim mas mesmo valor para todas as aplicações	Sim	
Latitude Geofence	Sim bloqueia o acesso	Não	
Longitude Geofence	Sim bloqueia o acesso	Não	
Network			
SSID	Sim mas mesmo valor para todas as aplicações	Existe opção mas não funcionou	Sim e dinâmico
BSSID	Sim mas mesmo valor para todas as aplicações	Existe opção mas não funcionou	Sime dinâmico
WIFI MAC	Sim mas mesmo valor para todas as aplicações	Sim	Sim e dinâmico
Estado Wifi	Sim	Não	Sim e dinâmico
Lista dos Pontos de Acesso (AP)	Sim bloqueia o acesso	Não	Sim bloqueia o acesso
Lista de Configurações de Rede	Sim bloqueia o acesso	Não	Sim bloqueia o acesso
Bluetooth MAC	Sim	Sim	Sim e dinâmico
Média			
Imagens	Existe opção mas não funcionou	Sim bloqueia o acesso	Sim bloqueia o acesso
Videos	Existe opção mas não funcionou	Sim bloqueia o acesso	Sim bloqueia o acesso
Audio	Existe opção mas não funcionou	Sim bloqueia o acesso	Sim bloqueia o acesso
Google			
Localização aleatória	Não	Não	Sim e dinâmico
Restringir acesso aos contactos	Não	Não	Sim bloqueia o acesso
Restringir acesso ao Calendário	Não	Não	Sim bloqueia o acesso
Restringir Marcadores e Histórico	Não	Não	Sim bloqueia o acesso

Capítulo 5

Contratos e Legislação Portuguesa

5.1 Termos de Contrato e Políticas da Google

O utilizador ao adquirir um dispositivo Android deverá criar uma conta Google de forma a poder usufruir das diversas funcionalidades que a Google proporciona, como por exemplo, o acesso ao Google Play. Para aceder a este serviço, o utilizador está sujeito aos termos e condições do contrato [30] e as políticas de privacidade [28].

A Google informa que recolhem diversas informações sobre o utilizador, para melhorarem os seus serviços *"de modo a apresentar anúncios e resultados de pesquisa mais relevantes, para o ajudar a estabelecer contato com as pessoas e a tornar a partilha com os outros mais fácil e rápida"* [28]. Informações nas quais destaco nome do utilizador, seu endereço de correio eletrónico, número de telefone, número de cartão de crédito, localização do dispositivo (por GPS, IP, outros sensores) e dados de navegação. Não serão dados a mais? Será que estes dados serão somente para os fins que o contrato informa? Pois não sabemos, confiamos e aceitamos as exigências contratuais da Google. Acredito que a maioria dos utilizadores comuns nem sequer lêem todos estes contratos e que tenham cada vez mais a sua vida exposta no mundo cibernético, sem estarem conscientes do perigo que podem estar sujeitos.

Perante os contratos [28] [30] o utilizador ainda não está apto para poder aceder ao Google Play, para tal, é necessário aceitar os termos e condições da utilização do Google Play [27]. No qual destaco, o **Ponto 2** em que informa da existência de conteúdos disponibilizados pela Google e outros por terceiros não filiados à Google, tal que, *"A Google não é responsável por quaisquer conteúdos no Google Play provenientes de outras origens, que não da Google, e não promove tais conteúdos"*. O **Ponto 3**, indica que a Google protege o utilizador contra aplicações maliciosas de terceiros e outros problemas de segurança recebendo informações sobre as ligações de rede do dispositivo, do sistema operativo e das aplicações de terceiros. No entanto, esta opção pode ser desativada pelo utilizador nas definições do Google no dispositivo. Por final, o **Ponto 8** relativamente às posições geográficas, em que o utilizador concorda em não apresentar informações falsas sobre a sua localização devido a existirem serviços que, só podem ser disponibilizados

somente para alguns países. O que é um problema quando utilizamos o IdentitySpoofing com as definições para "enganar" os serviços da Google.

Existe ainda, Contrato de Distribuição para Programadores do Google Play [29], o referente contrato é entre a Google e a terceira entidade que desenvolve aplicações e disponibiliza-as no Google Play. No qual destaco o **Ponto 4.3** *"O Utilizador concorda que, se utilizar a Loja para distribuir Produtos, protegerá a privacidade e os direitos legais dos utilizadores. Se os utilizadores lhe fornecerem, ou o Produto aceder ou utilizar, nomes de utilizador, palavras-passe ou outras informações de início de sessão ou pessoais, tem de avisar os utilizadores de que as informações estarão disponíveis para o Produto e fornecer o aviso de privacidade e a proteção legalmente adequados para esses utilizadores. Além disso, o Produto só poderá utilizar essas informações para os efeitos limitados para os quais o utilizador deu permissão..."*. Em análise todas as aplicações desenvolvidas por terceiros e disponibilizadas no Google Play deveriam ser confiáveis pois têm de proteger a privacidade e os direitos legais dos seus utilizadores.

5.2 Implicações na Legislação Portuguesa

É costume falar-se em "reserva da vida privada" e no **Ponto 1** do **Artigo 26.º** da Constituição da Republica Portuguesa [15] defende o direito "... à reserva da intimidade da vida privada e familiar e à proteção legal contra quaisquer formas de discriminação", de facto existem aplicações que não respeitam dos nossos direitos, como referido anteriormente, estas podem aceder à nossa localização, calendário, mensagens de texto, entre outras informações que divulgam em muito a intimidade da vida privada e familiar.

De igual forma, se o utilizador tirar uma fotografia do seu telemóvel, e esta foto para o próprio, pode ser algo pessoal, até mesmo privado, com certeza que não está interessado que essa informação seja acedida e transferida para servidores externos, ou até mesmo, para terceiras entidades, mesmo que o acesso e o tratamento dos dados seja feita com a maior das precauções e segurança.

O **artigo 70.ª** do do Código Civil [17] referênte à tutela geral da personalidade informa que a lei defende os indivíduos contra qualquer ofensa, tal que, *"... a pessoa ameaçada ou ofendida pode requerer as providências adequadas às circunstâncias do caso com o fim de evitar a consumação da ameaça ou atenuar os efeitos da ofensa já cometida."*.

O **artigo 80.ª** do Código Civil [17] referênte à reserva sobre a intimidade da vida privada defende que todos os cidadãos têm o direito à reserva da sua vida privada de outrem. No entanto, existem ações efetuadas por algumas das aplicações que não respeitam a nossa esfera pessoal, privada e secreta pois acedem e transmitem informação sensível sem qualquer tipo de consentimento do proprietário dos dados. Todos nós temos o direito de nos proteger em relação aos nossos costumes, hábitos, orientações sexuais, situação baáncaria e saúde.

A separação dos dados privados ou públicos na utilização de dispositivos inteligentes é

tudo menos binária, pois existem informações privadas. Num contexto em que poderá ser partilhada num grupo de amigos e não haverá qualquer tipo de abuso e a mesma informação sendo transmitida para entidades externas que poderá ser acedida por outros (humanos ou máquinas) poderão originar uma ofensa ao proprietário dos dados no caso de serem ou não revelados.

Ao analisarmos **Decreto de Lei n.º 67/98, de 26 de Outubro** [43], referente à proteção de dados pessoais no **artigo 2.º** defende que o *"O tratamento de dados pessoais deve processar-se de forma transparente e no estrito respeito pela reserva da vida privada...* e o **artigo 10.º** defende que o cliente deve ser informado das finalidades do seu tratamento. Nenhum destes artigos é efetivamente praticado por aplicações disponíveis no Google Play, pois não é com uma breve explicação sobre as permissões quando uma aplicação necessita de acesso a um recurso, que o utilizador é informado de como é feito o tratamento dos dados. Se aplicação efetua um pedido de acesso à localização do dispositivo a única informação fornecida ao utilizador é *"utiliza a localização do dispositivo"*.

É de extrema importância a análise do **Decreto de Lei 109/2009 de 15 de Setembro** [42] referente ao cibercrime. O **artigo 5.º** intitulado de sabotagem informática, em que o **Ponto n.º1** informa que sem permissão legal ou sem estar autorizado pelo proprietário do sistema informático quem perturbar o seu funcionamento através da introdução, transmissão, alteração, apagamento impedindo o acesso aos dados é punido que com pena de prisão de 5 anos ou com pena de multa até 600 dias. De certa forma, é um bom enquadramento das funcionalidades fulcrais da componente de *software* desenvolvida, pois para além de alteração do funcionamento do sistema impedindo o acesso legítimo aos dados às aplicações, a autorização por parte dos proprietários das aplicações não foi pedida. No entanto, esses dados são pessoais e os dados pessoais pertencem ao proprietário do dispositivo que tem o autoridade total tanto dos seus dados como do seu dispositivo.

Relativamente ao **artigo 7.º** intitulado de Interceção ilegítima, em que o **Ponto n.º1** informa que *"Quem, sem permissão legal ou sem para tanto estar autorizado pelo proprietário, por outro titular do direito do sistema ou de parte dele, e através de meios técnicos, intercetar transmissões de dados informáticos que se processam no interior de um sistema informático, a ele destinadas ou dele provenientes, é punido com pena de prisão até 3 anos ou com pena de multa"*. Podemos igualmente incorporar no funcionamento do IdentitySpoofing, embora a intersecção dos dados informáticos seja feita no dispositivo do proprietário e de forma a proteger a informação do utilizador e não, de forma a utilizar os próprios dados como uma fonte de informação.

Capítulo 6

Conclusão

A dependência dos telemóveis inteligentes está cada vez mais presente no estilo de vida das pessoas. É importante a melhoria contínua não só no desenvolvimento de novas funcionalidades mas igualmente dos mecanismos de proteção de informação sensível dos utilizadores.

A capacidade do Xposed realizar *hooks* ao sistema, todo o seu processo de instalação, configuração e utilização o torna mais robusto e vantajoso relativamente a todos os sistemas abordados, em que é necessário instalação de uma nova ROM.

A utilização da *framework* Xposed, foi essencial para o nosso objetivo proposto de criar um perfil falso do utilizador para cada uma das aplicações instaladas no dispositivo. Conseguindo assim, não só proteger informação do utilizador mas também gerar identificadores com inteligência e significado, de forma a criarmos a ilusão de uma identidade real e combater o *profiling* abusivo por parte de certas aplicações.

Relativamente às aplicações estudadas podemos afirmar que o IdentitySpoofing tem capacidades superiores devido à sua estratégia de lidar com pedidos de acesso à localização do dispositivo e identificadores únicos. Mas também, devido à sua interface amigável, em que o utilizador necessita apenas selecionar as restrições que deseja proteger e todos os processos são efetuados automaticamente e de uma forma dinâmica sem que este necessite de especificar qualquer tipo de informação manualmente.

O facto do IdentitySpoofing conseguir de "enganar" os serviços da Google é uma mais-valia, embora não seja uma opção aconselhada devido ao telemóvel ter de iniciar com configurações diferentes das originais.

Atualmente, os utilizadores comuns destes dispositivos computacionalmente poderosos sente-se seguros, muito devido a não estarem cientes das falhas existentes no mundo digital e do valor que têm os seus próprios dados.

A presente dissertação descreve ainda pontos interessantes em termos da legislação portuguesa enquadrada com controlo e ocultação de dados pessoais em sistemas informáticos e adverte à consciência das implicações legais que podemos estar sujeitos a implementar determinadas

soluções informáticas desta natureza, devido a manipulam o correto funcionamento do sistema e das aplicações instaladas no sistema.

É importante que a Google e toda a comunidade de programadores para Android continuem a ter precauções com a segurança e privacidade dos seus utilizadores, nota-se com o aparecimento do Android Marshmallow que o sistema de permissões melhorou bastante podendo os utilizadores revogar permissões impostas pelas aplicações, mas ainda existe um trabalho árduo de melhoria continua nesta área para ser desenvolvido.

6.1 Trabalho Futuro

Como futuro trabalho é importante a exploração de todos os conteúdos acedíeis através dos fornecedores de conteúdos, como por exemplo, os contactos, registo de chamadas, calendário, mensagens. Realizar um novo processo de *hook* para que, em vez de negar apenas o acesso às aplicações a esses conteúdos, consiga igualmente transmitir informação baseada em dados reais distintas das do utilizador. É importante referir que ao transmitirmos uma lista de contactos sem nenhum contacto ou não existir registos de chamadas de telefone no dispositivo pode levantar suspeitas relativamente à fidedignidade dos dados. No trabalho analisado em [10] descreve uma solução eficaz e coerente de como conseguir transmitir uma lista de contactos diferente da original e com dados realistas tendo em base a lista de contactos do utilizador

É fundamental explorar uma forma mais realista de enganar as aplicações relativamente à localização do utilizador, como por exemplo, gerar itinerários baseados em locais de interesse e posteriormente criar rotinas tendo em atenção os fins de semana e possíveis férias. Em [10] encontramos uma solução bastante interessante sem a necessidade de existir uma base de dados extra com as localizações de GPS, no entanto, seria necessário implementar uma solução que respondesse de forma eficiente aos pedidos de acesso à localização quando o dispositivo dispor do GPS desativo e os dados móveis ativos.

Seria igualmente interessante que todos os ficheiros enviados para o servidor fossem guardados em uma base de dados do tipo NoSQL, como por exemplo, a MongoDB [20], para temos os registos dos dados de cada dispositivo que pretendemos realizar os testes.

O servidor poderia efetuar operações de validação dos dados, e ao receber dados de localização efetuava uma pesquisa automaticamente em um serviço web para disponibilizar a sua representação gráfica.

Realizar mecanismos para que a base de dados fakedata poder ser partilhada e alterada por cada um dos seus utilizadores, assim conseguíamos ter uma renovação de dados para todos os utilizadores do IdentitySpoofing.

Bibliografia

[1] acpm.mobi. Genymotion xposed inspeckage. "https://acpm.mobi/genymotion-xposed-inspeckage/", 2016 (acedido Outubro 23, 2016).

[2] Seppo Alanärä. Backwards compatible, extended method to execute check number calculation of imei, May 30 2006. US Patent 7,054,657.

[3] Android. "https://developer.android.com/reference/android/provider/Settings.Secure.html", 2016 (acedido Agosto 19, 2016).

[4] Android. Content provider basics. "https://developer.android.com/guide/topics/providers/content-provider-basics.html/", 2016 (acedido Agosto 26, 2016).

[5] Android. Content resolver. "https://developer.android.com/reference/android/content/ContentResolver.html/", 2016 (acedido Agosto 26, 2016).

[6] Steven Arzt, Siegfried Rasthofer, and Eric Bodden. Susi: A tool for the fully automated classification and categorization of android sources and sinks. *University of Darmstadt, Tech. Rep. TUDCS-2013-0114*, 2013.

[7] GSM Association. Imei allocation and approval guidelines version 6.0. "http://www.gsma.com/newsroom/wp-content/uploads/2012/06/ts0660tacallocationprocessapproved.pdf", 2016 (acedido Agosto 19, 2016).

[8] Vitalii Avdiienko, Konstantin Kuznetsov, Alessandra Gorla, Andreas Zeller, Steven Arzt, Siegfried Rasthofer, and Eric Bodden. Mining apps for abnormal usage of sensitive data. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 1, pages 426–436. IEEE, 2015.

[9] Alastair R Beresford, Andrew Rice, Nicholas Skehin, and Ripduman Sohan. Mockdroid: trading privacy for application functionality on smartphones. In *Proceedings of the 12th workshop on mobile computing systems and applications*, pages 49–54. ACM, 2011.

[10] Daniel Borges. *Privacidade no Sistema Android*. MSc dissertação, Universidade de Aveiro, 2014.

[11] Jesse Burns. Developing secure mobile applications for android, 2008.

[12] P Cerwall, P Jonsson, et al. Ericsson mobility report: On the pulse of the networked society. *Ericsson, Jun*, 2015.

[13] CollegeDev. "https://github.com/CollegeDev/DonkeyGuard/blob/master/README.md", 2016 (acedido Agosto 12, 2016).

[14] CollegeDev. "https://github.com/CollegeDev/PDroid2.0_Stock", 2016 (acedido Agosto 12, 2016).

[15] VI REVISÃO CONSTITUCIONAL. Constituição da república portuguesa vi revisão constitucional [2004]. 1976.

[16] Terence Craig and Mary E Ludloff. *Privacy and big data*. "O'Reilly Media, Inc.", 2011.

[17] SUPREMO Tribunal de Justiça. Código civil português. "http://www.stj.pt/ficheiros/fpstjptlp/portugal_codigocivil.pdf", 2016 (acedido Agosto 29, 2016).

[18] XDA Developers. rovo89 profile. "http://forum.xda-developers.com/member.php?u=4419114/", 2016 (acedido Agosto 26, 2016).

[19] Joshua J Drake, Zach Lanier, Collin Mulliner, Pau Oliva Fora, Stephen A Ridley, and Georg Wicherski. *Android hacker's handbook*. John Wiley & Sons, 2014.

[20] Eliot Horowitz Dwight Merriman and Kevin Ryan. Mongodb unleashes the power of software and data for innovators everywhere. "https://www.mongodb.com/company/", 2016 (acedido Setembro 26, 2016).

[21] Enaikoon. Opencellid database. "http://opencellid.org/", 2016 (acedido Agosto 19, 2016).

[22] William Enck, Peter Gilbert, Seungyeop Han, Vasant Tendulkar, Byung-Gon Chun, Landon P Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N Sheth. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems (TOCS)*, 32(2):5, 2014.

[23] Genymotion. More than an emulater, a powerful virtualization platform to address all your professional needs. "https://www.genymotion.com/", 2016 (acedido Outubro 23, 2016).

[24] Geomidpoint. Random point generator. "http://www.geomidpoint.com/random/", 2016 (acedido Agosto 19, 2016).

[25] Google. Review app permissions thru android 5.9. "https://support.google.com/googleplay/answer/6014972?hl=en/", 2016 (acedido Agosto 26, 2016).

[26] Google. Google play. "https://play.google.com/store/", 2016 (acedido Agosto 26, 2016).

[27] Google. Termos de utilização do google play. "https://play.google.com/intl/pt-PT_pt/about/play-terms.htm/l", 2016 (acedido Agosto 26, 2016).

[28] Google. Privacidade e termos de utilização. "https://www.google.com/intl/pt-PT/policies/privacy/", 2016 (acedido Agosto 26, 2016).

[29] Google. Contrato de distribuição para programadores do google play. "https://play.google.com/intl/ALL_pt/about/developer-distribution-agreement.html/", 2016 (acedido Agosto 26, 2016).

[30] Google. Termos de serviço do google. "https://www.google.pt/intl/pt-BR/policies/terms/regional.html/", 2016 (acedido Agosto 26, 2016).

[31] Google. Google painel control. "https://www.google.com/settings/dashboard", 2016 (acedido Agosto 29, 2016).

[32] GrepCode. Android source code. "http://grepcode.com/search/?query=google+android&entity=project", 2016 (acedido Agosto 19, 2016).

[33] Grepcode. Cellidentitygsm. ""http://grepcode.com/file/repository.grepcode.com/java/ext/com.google.android/android/4.4.2_r1/android/telephony/CellIdentityGsm.java/", 2016 (acedido Agosto 26, 2016).

[34] Grepcode. Location. "http://grepcode.com/file/repository.grepcode.com/java/ext/com.google.android/android/4.2.2_r1/android/location/Location.java/", 2016 (acedido Agosto 26, 2016).

[35] Grepcode. Networkinfo. "http://grepcode.com/file/repository.grepcode.com/java/ext/com.google.android/android/4.2.2_r1/android/net/NetworkInfo.java/", 2016 (acedido Agosto 26, 2016).

[36] Grepcode. Telephonymanager. "http://grepcode.com/file/repository.grepcode.com/java/ext/com.google.android/android/4.4.2_r1/android/telephony/TelephonyManager.java/", 2016 (acedido Agosto 26, 2016).

[37] Grepcode. Wifimanager. "http://grepcode.com/file/repository.grepcode.com/java/ext/com.google.android/android/4.2.2_r1/android/net/wifi/WifiManager.java/", 2016 (acedido Agosto 26, 2016).

[38] Peter Hornyack, Seungyeop Han, Jaeyeon Jung, Stuart Schechter, and David Wetherall. These aren't the droids you're looking for: retrofitting android to protect data from imperious applications. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 639–652. ACM, 2011.

[39] Shine Jacob. "https://pragmaticparag.blogspot.pt/2012/07/generating-and-validating-number-using.html", 2016 (acedido Agosto 19, 2016).

[40] M66B. "https://github.com/M66B", 2016 (acedido Agosto 12, 2016).

[41] Anmol Misra and Abhishek Dubey. *Android security: attacks and defenses*. CRC Press, 2013.

[42] DIÁRIO DA REPÚBLICA. Lei do cibercrime lei n.º 109/2009 de 15 de setembro. "http://www.pgdlisboa.pt/leis/lei_mostra_articulado.php?nid=1137&tabela=leis", 2016 (acedido Agosto 29, 2016).

[43] DIÁRIO DA REPÚBLICA. Lei da protecção de dados pessoa lei n.º 67/98 de 26 de outubro. "http://www.segurancaonline.com/fotos/gca/l67_98_1262606544.pdf", 2016 (acedido Agosto 29, 2016).

[44] rovo89. "https://github.com/rovo89/XposedBridge/wiki/Development-tutorial/", 2016 (acedido Agosto 26, 2016).

[45] rovo89. "https://github.com/rovo89/XposedBridge/blob/master/src/de/robv/android/xposed/XposedBridge.java/", 2016 (acedido Agosto 26, 2016).

[46] rovo89. "http://forum.xda-developers.com/showthread.php?t=3034811", 2016 (acedido Setembro 26, 2016).

[47] Ricardo Di Lucia Santos. Redes gsm, gprs, edge e umts. "http://www.gta.ufrj.br/ensino/eel879/trabalhos_vf_2008_2/ricardo/1_2.html", 2016 (acedido Agosto 29, 2016).

[48] Statista. Number of apps available in leading app stores as of june 2016. "http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/", 2016 (acedido Agosto 26, 2016).

[49] Rohit Tamma and Donnie Tindall. *Learning android forensics*. Packt Publishing Ltd, 2015.

[50] TeamWin TWRP. "https://twrp.me/about/", 2016 (acedido Setembro 26, 2016).

[51] Virusshare. "https://virusshare.com/", 2016 (acedido Agosto 26, 2016).

[52] Yajin Zhou, Xinwen Zhang, Xuxian Jiang, and Vincent W Freeh. Taming information-stealing smartphone applications (on android). In *International conference on Trust and trustworthy computing*, pages 93–107. Springer, 2011.