Universidade Federal de Pernambuco

Centro de Informática

Graduação em Engenharia da Computação

Análise das Permissões e Violações de Privacidade em Aplicações para Android

Pedro Henrique Martins Barbosa

Trabalho de Graduação

Recife

Dezembro de 2017

Universidade Federal de Pernambuco
Centro de Informática

Pedro Henrique Martins Barbosa

Análise das Permissões e Violações de Privacidade em Aplicações para Android

Trabalho apresentado ao Programa de Graduação em Engenharia da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Engenharia da Computação.

Orientador: *Prof. Márcio Lopes Cornélio*

Recife
Dezembro de 2017

Ao meu avô, Sebastião de Souza Barbosa (in memorian).

Agradecimentos

Primeiramente, agradeço a Deus por me proporcionar saúde e força para vencer as dificuldades e pela oportunidade de concluir esse trabalho de graduação.

Aos meus pais e irmãos, que sempre me deram apoio e incentivo em todos os momentos da minha vida, dando assim a confiança necessária para lutar pelos meus objetivos. Agradeço também aos meus familiares sem os quais não poderia ter chegado onde estou.

A todos os meus professores do Centro de Informática (CIn), que ajudaram muito no meu aprendizado, transformando-me numa pessoa melhor. É um privilégio adquirir conhecimentos com pessoas dispostas a transformar o mundo.

Aos meus amigos, os quais tenho a sorte de tê-los em minha vida. Ao longo da graduação fizeram com que eu pudesse compartilhar tantos momentos maravilhosos. Agradeço também ao meu orientador Márcio Lopes Cornélio pelo suporte, incentivo e paciência, oferecendo todo o apoio para o desenvolvimento desse trabalho.

E finalmente a todas as pessoas que contribuíram para a minha formação acadêmica. Muito obrigado.

Resumo

A segurança no desenvolvimento de aplicativos é uma questão que preocupa grande parte dos desenvolvedores, sendo esta, do mesmo modo, muito importante também para os seus usuários. Ao fazer o download de um *app*, muitas permissões são solicitadas ao usuário para que se tenha acesso ao conteúdo do seu dispositivo: conexões, contatos, fotos, contas, dados, entre outros. Dessa forma, o Android possui um ótimo mecanismo de defesa: o seu sistema de permissões, que define uma série de ações que são ou não permitidas, podendo ou não violar a privacidade digital dos usuários. Neste trabalho, 41 aplicações Android, instaladas em um dispositivo móvel, e 10 aplicações *open source*, extraídas de repositórios do GitHub, foram investigadas. Todas elas foram analisadas levando em consideração parâmetros importantes, como o número de permissões e a quantidade de violações nas políticas de privacidade, sendo estas últimas mapeadas através de métodos API definidos. Para a análise do primeiro grupo de aplicações, foi desenvolvido um aplicativo Android (Check Permissions) com o objetivo de verificar se há variação no número de permissões. Enquanto isso, o segundo grupo de aplicações foi analisado utilizando-se o PoliDroid-AS, plugin do Android Studio, responsável pela detecção de desalinhamentos entre códigos-fonte e políticas de privacidade. Os experimentos realizados evidenciaram que permissões específicas são adicionadas ou retiradas (13 dos 41 aplicativos mudaram seu número de permissões) e que há muitas políticas de privacidade violadas nos códigos-fonte das 10 aplicações *open source*, demonstrando vulnerabilidades não transparentes aos usuários.

Palavras-chave: Aplicativos Android, permissões, políticas de privacidade, detecção de violação.

Abstract

Security in application development is a matter of concern to most developers, which is equally important for application users as well. When downloading an application, many permissions are required for the user to access the contents of their device: connections, contacts, photos, accounts, data, and more. In this way, Android has a great defense mechanism: their permissions system, which defines a series of actions that are or are not allowed to an application and may or may not violate users' digital privacy. In this work, 41 Android applications, installed on a mobile device, and 10 open source applications, extracted from GitHub repositories, were investigated. All of them were analyzed taking into account important parameters, such as the number of permissions and the amount of violations in the privacy policies, the latter being mapped through defined API methods. For the analysis of the first group of applications, it was developed an Android application (Check Permissions) in order to check if there is variation in the number of permissions. Meanwhile, the second group of applications was analyzed using PoliDroid-AS, an Android Studio plugin responsible for the detection of misalignments between source codes and privacy policies. Experiments have shown that specific permissions are added or removed (13 of 41 applications have changed their number of permissions), and there are many privacy policies violated in the source code of the 10 open source applications, demonstrating non-transparent vulnerabilities to users.

Keywords: Android applications, permissions, privacy policies, violation detection.

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Introdução

Conforme informações do jornal *The Independent*, atualmente existem mais dispositivos móveis do que pessoas. Enquanto que o número de indivíduos está em torno de aproximadamente 7,19 bilhões, esses aparelhos contabilizam cerca de 7,22 bilhões unidades [1]. Desse modo, no contexto de aplicativos para dispositivos móveis, há um número crescente de interações.

Aplicações móveis (aplicativos ou *apps*) são sistemas desenvolvidos para serem instalados em aparelhos móveis, tais como smartphones ou tablets, garantindo diversas funcionalidades [2]. Para operar, as aplicações podem exigir acesso aos recursos dos dispositivos em que estão instalados, bem como informações dos usuários contidas nesses aparelhos móveis.

Em contrapartida, à medida que os usuários instalam *apps*, seus aparelhos eletrônicos móveis podem produzir informações pessoais e dados, que vão desde gps, interações de mensagens de texto (SMS) ou localização, colocando em risco, por exemplo, a privacidade dos usuários.

As permissões são o mecanismo pelo qual os desenvolvedores revelam como suas aplicações irão interagir com os dispositivos dos usuários. Uma vez que a permissão é concedida, os aplicativos podem reunir informações do usuário.

De acordo com uma pesquisa do Pew Research Center [2], descobriu-se que 90% das pessoas têm conhecimento sobre como os aplicativos acessam ou usam seus recursos pessoais, sendo que 60% desses usuários optam por não baixar um aplicativo depois de descobrir a quantidade de dados pessoais que são por ele requisitados. Isso demonstra que os indivíduos estão preocupados com a informação exigida pelos seus aplicativos, mas não têm conhecimento sobre o que está acontecendo do outro lado da transação - as permissões e capacidades que os aplicativos são mais prováveis de pedir.

Outro ponto a se destacar são as políticas de privacidade que não são escritas pelos desenvolvedores, sendo essa tarefa atribuída a peritos legais em grandes empresas, uma vez que se tratam de documento legal. Entretanto, esses especialistas que elaboram as políticas de privacidade não estão familiarizados com o código-fonte do aplicativo, o que torna esse trabalho

difícil, uma vez que as relações entre linguagem natural e código-fonte nem sempre são óbvias e legais, dependendo dos escritores e desenvolvedores. Para pequenas empresas que não podem pagar conhecimentos legais, os desenvolvedores acabam ficando com a tarefa de escrever as políticas, levando-os a copiar as mesmas de outros aplicativos, o que pode ocasionar o desalinhamento.

É importante que os desenvolvedores de software não apenas estejam cientes de quais informações o seu aplicativo está coletando, mas também se sua política de privacidade é consistente com o que realmente está sendo coletado ou acessado.

O problema central do presente trabalho procura responder as seguintes questões: por que há vulnerabilidades em aplicações Android? Aplicativos modificam uso de permissões entre versões diferentes? A cada atualização de versão, a quantidade de permissões de aplicativos tem variação considerável? Alguma modificação não é explícita para o usuário? Em caso afirmativo, que modificações são essas?

1.1 Objetivos

Os objetivos do presente trabalho são:

- Apresentar um estudo sobre permissões e políticas de privacidade existentes em aplicações móveis, descrevendo as mais utilizadas e as principais diferenças entre elas;
- Criar uma aplicação Android para verificar a variação do número de permissões de aplicativos instalados em um dispositivo móvel a cada atualização de versões destes;
- Verificar possíveis violações de privacidade nos códigos-fonte Android em repositórios *open source*;

1.2 Estrutura do Documento

O restante deste documento está organizado da seguinte forma: o Capítulo 2 aborda a contextualização deste trabalho, apresentando uma revisão bibliográfica sobre o sistema Android,

versões do sistema, suas características e arquitetura de segurança, assim como o funcionamento do seu sistema de permissões. Além disso, expõe a definição de política de privacidade e ontologia.

O Capítulo 3 refere-se a metodologia utilizada para analisar as permissões de aplicações em um dispositivo real e detectar violações de privacidade em códigos-fonte de aplicações do GitHub. O Capítulo 4 apresenta os resultados da análise realizada ao decorrer da produção do presente trabalho, descrevendo quais são as permissões e violações de privacidade mais frequentes nas aplicações escolhidas. Por último, o Capítulo 5 expõe as considerações finais deste trabalho, além de sugestões para projetos futuros.

CAPÍTULO 2

Fundamentação Teórica

Este capítulo apresenta conceitos básicos necessários ao entendimento deste trabalho. Inicialmente é apresentado o sistema operacional Android, detalhando suas versões e seus principais elementos. Em seguida, são apresentadas as permissões e os grupos de permissões desse sistema. Por fim, exibimos os conceitos de políticas de privacidade e ontologias com o uso de linguagem natural.

2.1 Sistema Operacional Android

O sistema Android foi apresentado à sociedade no final de 2007 através da Open Handset Alliance (OHA) para fornecer uma plataforma aberta que oferecesse liberdade de inovação no mercado de dispositivos móveis. Hoje, a OHA é formada por um grupo de 84 empresas de tecnologia [3], lideradas pelo Google, que se uniram para acelerar a inovação nos aparelhos móveis, fornecendo uma melhor experiência com menor custo aos usuários desses dispositivos [4]. Dentre essas empresas, há:

- **Empresas de comercialização**: Accenture, Intrinsyc, Aplix Corporation, Borqs, entre outras;
- **Fabricantes de chips**: Qualcomm, Intel, Nvidia, Arm, entre outros;
- **Fabricantes de celulares**: Samsung, Dell, Motorola, LG, Huawei, entre outros;
- **Empresas de software**: Google, EBay, entre outras;
- **Operadoras de telefonia**: Vodafone, Telefónica, TIM, T-Mobile, Sprint, entre outras.

O Android atualmente é a plataforma móvel mais popular, estando presente em mais de 190 países ao redor do mundo, sendo adicionados, a cada dia, mais de um milhão de novos

dispositivos móveis a esse montante [5]. Esse sistema operacional é baseado no *kernel* do Linux que incorpora bons recursos de segurança, cooperando com desenvolvedores para manter o Android e o seu ecossistema seguros. Tal fato permite um ótimo ecossistema de *apps* e dispositivos construídos ao redor da plataforma Android, que disponibiliza um poderoso framework, além de várias ferramentas e recursos que favorecem seu processo de difusão, tais como:

- Projeto de Código Aberto do Android (Android Open Source Project – AOSP), que disponibiliza o código-fonte do Sistema Operacional Android sob a licença Apache 2.0;
- Android SDK juntamente com a IDE Android Studio, que contém as ferramentas e recursos para o desenvolvimento de aplicações Android;
- Google Play, que é uma loja virtual que fornece um ambiente para que os desenvolvedores possam publicar suas aplicações para os usuários do Android;
- Outras ferramentas como o ProGuard, Emulator, AVD Manager, logcat, adb.

2.2 Versões do Android

As várias versões lançadas do Android, mostradas na Figura 2.1, trazem suas vantagens e desvantagens que foram determinantes para a evolução do sistema. É interessante verificar que todas as versões recebem nomes de doces em ordem alfabética, onde as letras A e B seriam as versões 1.0 e 1.1, não lançadas ao público, supostamente chamadas de Apple Pie e Banana Bread.

A seguir será mostrada cada uma delas e assim será possível entender o que melhorou de uma para outra [6].

- **Android 1.0**: Conhecido como *Alpha*, a primeira versão comercial do sistema Android, foi lançada em setembro de 2008 junto com a chegada do primeiro dispositivo a ser equipado com esse sistema operacional, o HTC Dream. Os principais recursos incorporados a esta versão foram: aplicação *Android Market* que realiza downloads e atualiza aplicativos através do aplicativo *Market*; *Web Browser* para exibir, dar zoom e suporte a páginas em HTML e XHTML; notificações na barra de status; suporte à câmera, contudo

nesta versão não havia opções de modificar a resolução da câmera, qualidade, balanço de branco, entre outros.

- **Android 1.1**: Conhecido como *Beta*, foi lançado em fevereiro de 2009, sendo a primeira atualização comercial do sistema Android. Os principais recursos adicionados foram: suporte para letreiros em layouts; tempo de limite de tela maior para chamadas padrão feitas pelo viva-voz; capacidade de salvar anexos de mensagens; comentários e mais detalhes sobre diversos assuntos pesquisados no Google Maps.

- **Android 1.5**: Conhecido como *Cupcake*, foi lançado em abril de 2009 baseado no Kernel Linux 2.6.27, produzindo várias melhorias para o sistema operacional como na parte do GPS, câmera, upload de vídeos e fotos para o YouTube, entre outros. Porém, a principal novidade foi o lançamento do primeiro Android com o *touch screen* e o teclado virtual. Também surgiu a ideia de *widgets* - pequenos aplicativos que são executados na tela inicial.

- **Android 1.6**: Conhecido como *Donut*, foi lançado em setembro de 2009 baseado no kernel Linux 2.6.29, trazendo pequenas mudanças para o sistema. Nesta versão, o sistema fala e escuta através da conversão de texto em voz (TTS) e conversão de voz em texto (STT). Com o auxílio das pesquisas de voz, a *home* do Android garantiu mais funcionalidades, facilitando o uso do sistema operacional pelo usuário. Além da API de programação para uso de *text-to-speech*, os principais recursos desta versão foram: interfaces para a programação de aplicativos com reconhecimento de gestos; suporte para resoluções de tela (320×240 e 800×480); capacidade de selecionar vários recursos simultaneamente.

- **Android 2.0 e 2.1**: Conhecido como *Eclair*, foi lançado em outubro de 2009 e posteriormente atualizado em janeiro de 2010. Trouxe diversas melhorias ao sistema, destacando a parte das câmeras e mapas. Nessa versão foi projetado o suporte ao HTML5 e a múltiplas contas do Google e sincronização.

- **Android 2.2**: Conhecido como *Froyo*, foi lançado em maio de 2010 baseado no kernel Linux 2.6.32, trazendo melhorias de desempenho para o sistema operacional, como o compilador JIT (Just in Time). Nessa versão foram acrescentados os seguintes recursos: Wi-Fi Hotspot (compartilhamento de internet com computadores com hotspots para até

8 dispositivos) e USB Tethering; suporte ao flash; introdução da barra fixa de atalhos: navegador, telefone e launcher; *Android Market* pode atualizar aplicativos automaticamente. Além de que os desenvolvedores puderam criar apps que aprimoram a segurança do aparelho, como as telas de bloqueio e a possibilidade de armazenar aplicativos no cartão SD.

- **Android 2.3**: Conhecido como *GingerBread*, foi lançado em dezembro de 2010 baseado no kernel Linux 2.6.35, trazendo novidades na câmera, possibilitando alternar entre a câmera frontal e traseira. Além disso, apresentou melhorias na funcionalidade de *copy-paste*, excelente ganho com o gerenciamento de bateria, suporte à NFC (*Near Field Comunicattions*) e aos sensores de movimento.

- **Android 3.0**: Conhecido como *Honeycomb*, foi lançado em fevereiro de 2011 baseado no kernel Linux 2.6.36, com o sistema operacional focado nos tablets e aprimorando a experiência dos usuários. Entre as novidades desta versão, se evidenciam: barra de ação, permitindo acessar navegação, widgets e outros tipos de conteúdo; interface remodelada e novo sistema multitarefas.

- **Android 4.0**: Conhecido como *Ice Cream Sandwich*, foi lançado em outubro de 2011 baseado no kernel Linux 3.0.1. Nesta versão, várias novidades foram adicionadas tanto no funcionamento quanto no design. Um dos objetivos desta versão do Android era unificar as interfaces separadas do Android para smartphones (Gingerbread) e para tablets (Honeycomb), o que foi realizado com sucesso. Além disso, a empresa aprimorou as "diretrizes para interface humana" do Android, ou seja, simplificou e modernizou a experiência global com o sistema operacional. Adicionou, ainda, novos recursos ao Android Ice Cream Sandwich, como a nova tela inicial e o suporte à tecnologia NFC.

- **Android 4.1**: Conhecido como *Jelly Bean*, foi lançado em junho de 2012 baseado no kernel Linux 3.0.31. Nesta versão, houve ganhos com relação ao desempenho do sistema, trazendo uma interface renovada e notificações expansíveis. Além disso, incluiu a tecnologia Photo Sphere, para produção de imagens em 360º e a possibilidade de realizar gestos na tela de bloqueio para acessar rapidamente a câmera do celular.

- **Android 4.4**: Conhecido como *KitKat*, foi lançado em outubro de 2013, elevando o

desempenho do sistema, otimizando a memória e aperfeiçoando a tecnologia touchscreen a respostas mais rápidas e com maior precisão. Além disso, trouxe aperfeiçoamentos no Bluetooth, Print Framework, sensores, NFC. Nesta versão, também foi criada a API de Transitions para animações personalizadas.

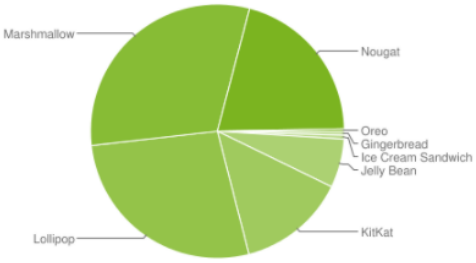
- **Android 5.0 e 5.1**: Conhecido como *Lollipop*, foi lançado em novembro de 2014. O Lollipop trouxe várias novidades e uma nova política visual, denominada de Material Design, na qual passou a guiar não apenas o Android, mas todos os produtos da empresa, inclusive os serviços web. Além disso, houve melhorias nas notificações, que aparecem também na tela de bloqueio (*Lock Screen*) e as head-up notificações, que aparecem no topo da tela com alta prioridade. Outra novidade desta versão foi o projeto Volta, que trouxe ferramentas para auxiliar a análise do uso da bateria nos *apps*.

- **Android 6.0**: Conhecido como *Marshmallow* ou Android M, foi lançado em outubro de 2015. Dentre as principais novidades desta versão, pode-se citar a inclusão de um novo modelo de permissão no qual apenas oito categorias precisam ser ativadas na primeira vez em que o usuário usa o aplicativo. Sendo assim, elas deixam de ser concedidas automaticamente no momento em que o usuário faz o download. Além disso, oferece suporte nativo para o reconhecimento de impressões digitais e também apresenta um novo sistema de gestão de energia, chamado “Doze”.

- **Android 7.0 e 7.1**: Conhecido como *Nougat* ou Android N, foi lançado em agosto de 2016. As principais características desta versão são: encriptação nativa no qual smartphones encriptados vão funcionar mesmo após reiniciação inesperada; API JobScheduler tornando o smartphone mais rápido; uso da tecnologia Vulkan.

- **Android 8.0**: A nova versão do SO Android, conhecido como *Android Oreo*, foi lançado na Google I/O em agosto de 2017. Apresenta as seguintes novidades: facilidade em acompanhar o consumo de bateria por parte dos aplicativos e também pelos próprios serviços do Android; o recurso de conversão de texto em fala, adquiriu melhorias, como o suporte a outros idiomas além do nativo do sistema.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.5%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.5%
4.1.x	Jelly Bean	16	2.2%
4.2.x		17	3.1%
4.3		18	0.9%
4.4	KitKat	19	13.8%
5.0	Lollipop	21	6.4%
5.1		22	20.8%
6.0	Marshmallow	23	30.9%
7.0	Nougat	24	17.6%
7.1		25	3.0%
8.0	Oreo	26	0.3%



Dados coletados durante um período de 7 dias encerrado em 2017/9/11.
Todas as versões com menos de 0,1% de distribuição não foram exibidas.

Figura 2.1 Versões do sistema operacional *Android*
Fonte: Android Developers [7]

2.3 Arquitetura de Segurança

A arquitetura geral do Android, ilustrada na Figura 2.2, é padronizada e desenvolvida em camadas. As camadas são: aplicações Android; Framework da aplicação, que gerencia o ciclo de vida dos componentes Android e sua intercomunicação; bibliotecas do núcleo Java, que servem para as aplicações e para o Framework Android; máquina virtual, que executa aplicações Android nos dispositivos móveis; bibliotecas nativas escritas em C/C++, que são responsáveis por manipular diferentes tipos de dados; e a camada base, que é o núcleo (*kernel*) do Linux.

- **Aplicações:** Essa camada, que está no topo da pilha de software do Android, abriga todos os *apps* que são executados sobre o sistema operacional Android, tais como: Mapas, *browser* (navegador de internet), SMS/MMS, cliente de email, alarme, álbum de fotos, dentre outros;

- **Framework Android:** Nessa camada ficam alguns blocos com os quais as aplicações interagem diretamente. Esses programas gerenciam funções básicas como: gerencia-

mento de recursos (Resource Manager), gerenciamento de localização (Loaction Manager), gerenciamento de atividades (Activity Manager), provedores de conteúdo (Content Providers), gerenciamento de telefone (Telephony Manager), gerenciamento de pacote (Package Manager), entre outros.

- **Bibliotecas:** Camada que tem as bibliotecas C/C++ utilizadas pelo sistema, além das bibliotecas de multímidia, funcionalidades para navegadores web, aceleração de hardware, funções para gráficos, renderização 3D, dentre outros;

- **Android Runtime:** Nessa camada é instanciada a máquina virtual Dalvik para execução de cada aplicativo do Android. Essa máquina virtual é a melhor referente a desempenho, maior integração com o novo hardware e é projetada para executar vários processos paralelamente;

- **Kernel do Linux:** Nessa camada ocorrem os acessos de hardware do sistema Android, fornecendo alguns aspectos importantes ao seu modelo de segurança, como por exemplo o modelo de permissões aos arquivos baseado no usuário; isolamento de processos, onde cada processo tem endereçamento, instruções, estado e recursos próprio, tal como uma máquina virtual; mecanismo de comunicação inter processos, onde um processo só se comunica com outro através de mecanismos bem definidos; e por fim, a possibilidade de remoção de partes desnecessárias do kernel.

2.4 Permissões do Sistema

Cada aplicativo do Android, quando é executado, possui uma identidade diferente de grupo e de usuário do Linux. Partes do sistema também estão em identidades distintas, onde o Linux, isola os aplicativos uns dos outros e do próprio sistema. Portanto, cada aplicativo funciona em uma *sandbox* de processo, na qual devem compartilhar explicitamente seus recursos e informações. Para os recursos não fornecidos pela *sandbox* básica são declaradas as permissões do sistema. Em seguida, os aplicativos declaram estaticamente as permissões que requerem e o sistema Android solicita a autorização do usuário.

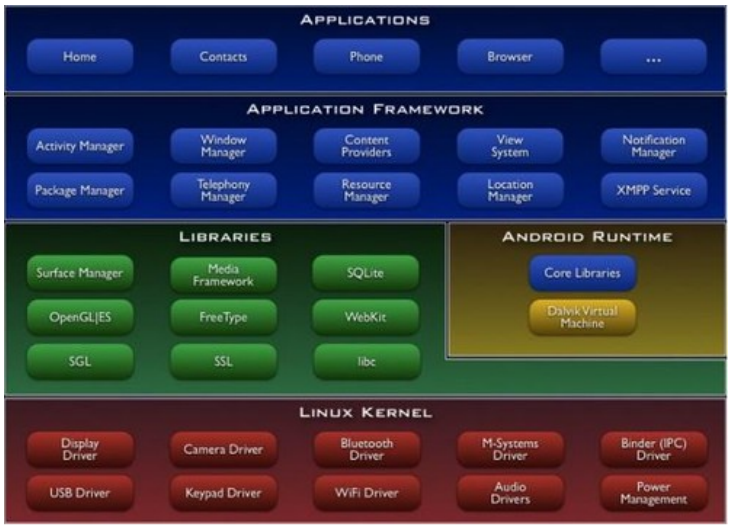


Figura 2.2 Camadas da arquitetura do sistema operacional *Android*

Fonte: [5]

2.4.1 Permissões Normais e Perigosas

A partir do Android 6.0, os usuários começaram a ter mais controle sobre os recursos de um aplicativo, uma vez que as permissões são concedidas quando o aplicativo está em execução. Diferentemente das versões anteriores, onde as permissões são concedidas quando o *app* é instalado. Dessa forma, as permissões do sistema são divididas em vários níveis de proteção, sendo os mais importantes a serem considerados: permissões normais e perigosas [7].

- **Permissões Normais:** Abrangem áreas onde o *app* precisa acessar dados/recursos fora da sandbox do aplicativo, porém apresenta pouco risco à privacidade do usuário ou à operação de outros aplicativos.

- **Permissões Perigosas:** Abrangem áreas onde o aplicativo precisa de dados/recursos que envolvem informações pessoais do usuário, podendo afetar os dados armazenados deste ou a operação de outros aplicativos. Por exemplo, ler os contatos do usuário é uma permissão perigosa. Se uma aplicação declara que necessita de uma permissão perigosa, o usuário precisa concedê-la explicitamente ao aplicativo.

2.4.2 Grupos de Permissões

As permissões perigosas do sistema Android pertencem a grupos de permissão, conforme pode ser visualizado na Tabela A.1. Caso um dispositivo esteja executando o Android 6.0 (*Marshmallow*), quando o aplicativo solicita uma permissão perigosa, o sistema adotará um dos seguintes comportamentos [7]:

- Caso um *app* solicite uma permissão perigosa listada no seu arquivo AndroidManifest.xml e não tem nenhuma permissão no grupo de permissões, o sistema mostrará uma caixa de diálogo ao usuário, descrevendo o grupo de permissões que o aplicativo quer acessar.

- Caso um *app* solicite uma permissão perigosa listada no seu AndroidManifest.xml e a aplicação já tem outra permissão perigosa no mesmo grupo de permissões, o sistema concederá a permissão sem nenhuma interação com o usuário.

Qualquer permissão pode pertencer a um grupo de permissões, inclusive as normais e as definidas por aplicativo. Entretanto, os grupos de permissões apenas influenciam a experiência do usuário caso trate-se de uma perigosa. Conclui-se, portanto, que é possível ignorar um grupo de permissões que apresente apenas permissões normais.

2.4.3 Uso de Permissões

Os aplicativos não têm permissões associadas por padrão, o que significa que não podem fazer nada que prejudique a experiência do usuário ou as informações naquele dispositivo. Para usar os recursos protegidos do aparelho móvel, é preciso incluir uma ou mais tags <uses-permission> no AndroidManifest.xml do aplicativo [7].

Por exemplo, um app que precisa de acesso a internet especificaria da seguinte forma:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.app.myapp">
    <uses-permission android:name="android.permission.INTERNET" />
    ...
</manifest>
```

Desse modo, caso um aplicativo liste permissões normais no seu arquivo AndroidManifest.xml, o sistema automaticamente irá concedê-las. Mas, se este aplicativo lista permissões perigosas, o sistema solicita ao usuário para concedê-las explicitamente. A forma como o Android faz estas solicitações depende da versão do sistema que o aplicativo necessite [7].

2.5 Content Providers

Content Provider (Provedor de Conteúdo) é um componente de aplicação que compartilha dados com outras aplicações. Pode especificar permissões que outros *apps* devem ter para acessar os dados do provedor, garantindo que o usuário saiba quais dados um aplicativo tentará acessar. Portanto, outros *apps* solicitam as permissões que precisam para acessar o provedor e os usuários veem as permissões solicitadas quando instalam o aplicativo.

Caso um aplicativo do provedor não especifique nenhuma permissão, outros *apps* não terão acesso aos dados do provedor. No entanto, os componentes no aplicativo do provedor sempre têm acesso de leitura e gravação, independentemente das permissões especificadas, enquanto que outros podem atribuir-lhes URIs específicos e outros aplicativos para que possam funcionar.

Para obter as permissões para acessar um provedor, um aplicativo as solicita como um elemento <uses-permission> no arquivo de AndroidManifest.xml. Quando o *Android Package Manager* instala o aplicativo, o usuário precisa aprovar todas as permissões que o aplicativo solicita. Se o usuário aprovar todas, o gerente de pacotes continuará a instalação, mas se o usuário não as aprovar, o gerente de pacotes interromperá a instalação.

A finalidade de conferir ao provedor de conteúdo a capacidade de conceder acesso aos dados deve ser bem clara. Um exemplo são os anexos em um aplicativo de e-mail, onde esse acesso precisa ser protegido pelas permissões visto que os dados de usuário são confidenciais. Porém, se um determinado URI para um anexo de imagem for concedido a um visualizador de imagens, esse não terá a permissão para abrir o anexo, pois não há motivo para reter uma permissão para acessar os e-mails.

Para resolver esse problema é importante usar as permissões por URI. Ao inicializar uma atividade ou retornar um resultado para uma atividade, é necessário que seja definido ao menos um dos seguintes comandos:

- Intent.FLAG_GRANT_READ_URI_PERMISSION
- Intent.FLAG_GRANT_WRITE_URI_PERMISSION

Isso concede à permissão de atividade recebida o acesso a um URI de dados específicos na *intent* [8] - objeto de mensagem que pode ser usado para solicitar uma ação de outro componente de aplicativo -, independentemente se ele tem ou não a permissão para acessar os dados no provedor de conteúdo correspondente à intent.

Portanto, essa solução permite a interação do usuário com a aplicação conduzindo privilégios ad-hoc de permissão otimizadas. Esse pode ser um recurso fundamental para reduzir as permissões necessárias a aplicativos somente às que são diretamente relacionadas ao comportamento. Para conceder permissões URI otimizadas, no entanto, é preciso certa cooperação com o provedor de conteúdos que retém esses URI's.

Recomenda-se que os provedores de conteúdos implementem essa unidade e declarem que são compatíveis com ela pelo atributo Android grantUriPermissions ou pela tag <grant-uri-permissions>:

```
<provider android:name="com.phmb2.apps.notes.providers.
    NotesContentProvider"
            android:authorities="com.phmb2.apps.notes.providers.
                NotesContentProvider"
            android:grantUriPermission="true"
            <grant-uri-permission android:pathPattern="/notes/" />
</provider>
```

2.6 Políticas de Privacidade

Além das permissões de sistemas padrão para acesso à API documentadas - rotinas e padrões de programação para acesso a um aplicativo de software - em arquivos AndroidManifest.xml, há as políticas de privacidade das aplicações móveis. São elas que averiguam quais informações são coletadas e usadas pelos aplicativos.

Essas políticas são importantes porque servem como o principal meio de comunicação com os usuários, informando-os quais e como informações pessoais confidenciais são utilizadas.

Sobre isto, as empresas de aplicativos publicam suas políticas de privacidade e os usuários acessam-nas, com a finalidade de decidir se irão aceitar ou não os termos estipulados antes de instalar os aplicativos [9].

Entretanto, a maioria dessas políticas são difíceis de entender por possuírem uma natureza ambígua, o que pode levar os usuários a ignorarem políticas de leitura, mesmo que tenham preocupações sobre práticas de coleta de informações.

Além disso, os desenvolvedores podem não conseguir cumprir as políticas de privacidade corretamente. Um obstáculo importante na compreensão e análise da privacidade é que não existe um formato padrão para apresentar a informação. A linguagem, a organização e os detalhes das políticas podem variar de aplicação para aplicação.

2.7 Ontologias e Processamento de Linguagem Natural

Ontologias são técnicas de organização de dados que vêm ganhando atenção nos últimos anos, principalmente no que diz respeito à representação formal de conhecimento [10]. Criadas por especialistas, as ontologias têm sua estrutura baseada na descrição de conceitos e dos relacionamentos semânticos entre eles, além de gerarem uma especificação formal e explícita de uma conceitualização compartilhada.

O crescimento do uso das ontologias existe devido à promessa de compartilhamento e entendimento comum de algum domínio de conhecimento que possa ser comunicado entre pessoas e computadores. Neste sentido, ontologias têm sido desenvolvidas para facilitar o compartilhamento e reutilização de informações [11].

Podem ser utilizadas, por exemplo, em Processamento de Linguagem Natural (PLN) [12], que se trata de uma área da Inteligência Artificial (IA) que estuda a capacidade e as limitações de uma máquina em entender a linguagem dos seres humanos. PLN fornece aos computadores a capacidade de entender e compor textos em que “entender” um texto significa reconhecer o contexto, realizar análise sintática, semântica, léxica e morfológica, extrair informações, interpretar os sentidos, analisar sentimentos e até mesmo aprender conceitos com os textos processados.

CAPÍTULO 3

Metodologia

O objetivo deste trabalho é realizar uma análise das permissões de aplicativos instalados em um dispositivo móvel e também verificar a ocorrência de violações de privacidade em aplicações Android *open source*. Com isto, será possível identificar quais as permissões e violações que estão mais evidentes atualmente nos aplicativos, ajudando assim os desenvolvedores a melhorar suas técnicas de implementação de *apps*, levando em consideração a segurança dos dados dos usuários.

A presente análise foi realizada em algumas etapas: (i) definição das permissões e de grupos de permissões do sistema operacional Android e critérios de análise, (ii) seleção de aplicações de dispositivos móveis, (iii) seleção de códigos de aplicações *open source*, (iv) extração de dados e análise. Os resultados dessa análise serão apresentados no Capítulo 4.

Há alguns trabalhos que descrevem análises de permissões e detecção de violações de privacidade. O Pew Research Center[13] que investigou mais de 1 milhão de aplicativos disponíveis na *Google Play Store*. Nesta pesquisa foram coletados vários dados sobre as permissões dos *apps*, nos quais constatou-se que usuários não instalam um aplicativo quando descobrem quantos dados pessoais este requer para utilizá-lo. A maioria das permissões buscam acesso ao hardware de um dispositivo em vez de informações pessoais de um usuário e as permissões de aplicativos mais comuns permitem o acesso à conectividade de internet de um smartphone.

Já o trabalho desenvolvido por pesquisadores da University of Texas e Carnegie Mellon University apresentou um estudo sobre como detectar violações de privacidade em aplicações Android utilizando um framework baseado em uma ontologia de frases de políticas de privacidade e uma coleção de mapeamento de métodos de API para tais frases [14].

Todavia, esses dois trabalhos mostram apenas informações gerais sobre as permissões e violações que ocorrem em determinados *apps* e repositórios do GitHub, não detalhando o porque existem vulnerabilidades em aplicações Android e o quanto isso afeta negativamente a qualidade de um software.

Desta forma, é necessário entender se há uma variação considerável de permissões a cada atualização de versões de um aplicativo e possíveis vulnerabilidades ocasionadas por violações de privacidade.

3.1 Ferramentas Utilizadas

Para realização das análises das permissões e das políticas de privacidade foram utilizadas duas ferramentas: Android Studio e o PoliDroid.

3.1.1 Android Studio

O Android Studio é um ambiente de desenvolvimento integrado para desenvolvimento Android baseado no software IntelliJ IDEA da JetBrains. Esse ambiente foi utilizado para a implementação da aplicação Check Permissions criada para coletar os dados gerados pela análise de permissões de alguns aplicativos da *Google Play Store* instalados em um aparelho móvel. Possui as seguintes propriedades:

- Possui ferramentas para capturar performance, usabilidade e compatibilidade de versão;
- Suporte para compilações baseadas em Gradle (ferramenta de automação de compilação que facilita o carregamento de APIs externas);
- Tem refatoração para Android e reparações rápidas;
- Permite a integração com a ferramenta ProGuard para melhorar a segurança e desempenho no código do aplicativo;
- Possui um editor de layout que permite aos usuários arrastarem componentes de interface com a opção de pré-visualizar layouts em várias configurações de tela;

3.1.2 PoliDroid

O Polidroid é um framework que detecta inconsistências entre métodos dos códigos-fonte de aplicativos Android e suas políticas de privacidade correspondentes. Para esse trabalho, foi

utilizado o PoliDroid-AS, plugin do PoliDroid [9] [15] integrado com o Android Studio. Esse plugin utiliza análise estática e processamento de linguagem natural para identificar relações entre código e linguagem natural. O PoliDroid-AS foi construído com base em uma coleção de ferramentas, apresentando uma arquitetura que permite a verificação de consistência de privacidades em tempo real, como visto na Figura 3.1.

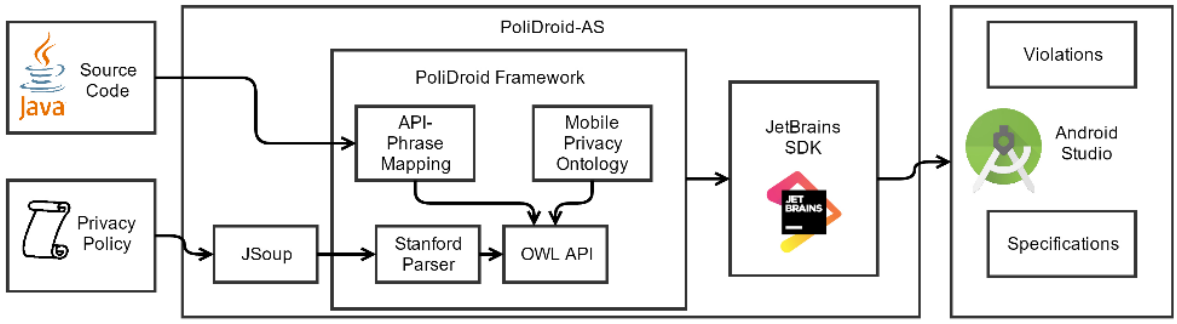


Figura 3.1 Arquitetura do PoliDroid-AS

Fonte: [9]

Funciona em um aplicativo por vez e, como as políticas de privacidade são incluídas com outros documentos legais - tais como os termos de serviço -, muitas vezes exibidos na página da aplicação ou no próprio *Google Play Store*, o PoliDroid-AS recebe como entrada um documento em texto ou um arquivo HTML incluindo a política de privacidade.

Utiliza o mapeamento e a ontologia, respectivamente, para verificar desalinhamentos no código-fonte. Como um exemplo de conjunto de mapeamento, pode-se considerar a Tabela 3.1. Para o mapeamento, os dados estão em um arquivo de valores (CSV) como entrada, contendo frases de políticas de privacidade na primeira coluna e métodos de API na segunda. Já para a ontologia é utilizado um arquivo de linguagem de Ontologia da Web (OWL) [16].

Se a entrada estiver no formato HTML, o PoliDroid-AS convoca o Jsoup - biblioteca Java de análise de HTML - para tirar as *tags* HTML e produz uma versão em texto para a entrada, sendo este filtrado até os parágrafos de coleta de dados relevantes para a privacidade. Em seguida, todas as frases no resultado são convertidas em árvores de análise, usando o analisador Stanford CoreNLP5, no qual a análise é aplicada para remover frases de polaridade negativa. Por fim, os componentes que foram reduzidos são comparados com os conceitos de ontologia de privacidade móvel para encontrar uma partida usando a API Web-Ontology Language (OWL)

e o PoliDroid-AS procura por frases no resultado dos parágrafos para os quais os métodos da API são mapeados.

Como o PoliDroid-AS foi construído usando o JetBrains IDE Plugin SDK6, para que ele possa interagir diretamente com o IntelliJ-based IDEs, seu componente central é uma extensão da ferramenta de inspeção do IntelliJ, que permite visitar todas as chamadas de método da API dentro do código-fonte que está sendo desenvolvido dentro do IDE e verificar se a chamada do método da API é representada pela política de privacidade do *app*, como pode ser visto na Figura 3.2.

Tabela 3.1 Exemplo Mapeamento Frase - Método API

Frases	Método API
gps information	getSpeed()
ip address	getIP()
unique identifier	getIP()

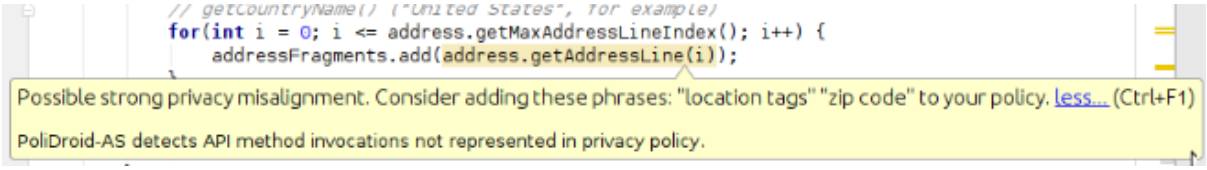


Figura 3.2 Exemplo de desalinhamento no código-fonte

3.2 Critérios de Análise

O que se pretende analisar com a coleta dos dados são permissões aplicadas em cada aplicativo e as violações de privacidade detectadas em códigos-fonte de aplicações Android que estão em repositórios do GitHub. Portanto, será possível verificar quais permissões, grupos de permissões e políticas de privacidade apresentadas no Capítulo 2 estão sendo mais utilizadas no desenvolvimento de software móvel Android.

3.3 Seleção de Aplicativos de Dispositivos Móveis

Para a análise de permissões foi utilizado um smartphone Motorola G4 Plus com o sistema operacional Android 6.0.1. Os aplicativos selecionados para esta análise, instalados no dispositivo móvel, são os seguintes: Agenda, Ajuda do Dispositivo, Alerte Me, App Box, BB, Calculadora, Câmera, Check Permissions, Chrome, CittaMobi, Configurar, Contatos, DFNDR Security (antigo PSAFE DFNDR), Documentos, Downloads, Drive, Facebook Lite, Fotos, Gmail, Google, Google Play Filmes e TV, Google Play Música, Google Play Store, Hangouts, Itaú, Maps, Mensagens, Moto, Notificações Motorola, Nubank, Nuo, Pluto, Radio FM, Serviços TIM, Slack, Snapchat, Telefone, Tinder, Uber, WhatsApp e YouTube. Alguns deles podem ser visualizados na Figura 3.3.

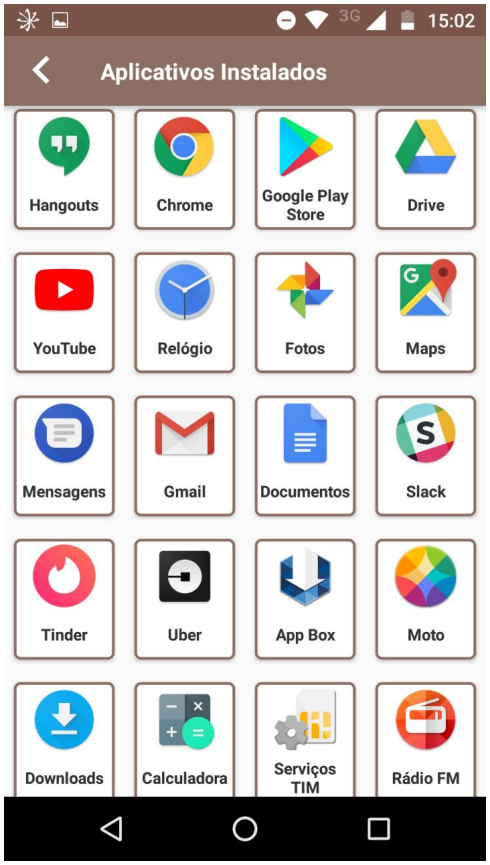


Figura 3.3 Tela Aplicativos Instalados

Muitos desses aplicativos foram escolhidos por serem bem conhecidos, e consequentemente apresentarem muitos usuários, para quem as permissões são definidas. Como pode ser visto na

Figura 3.4, dos 41 *apps* selecionados, 39 foram classificados como *Misc Permisssions* (*apps* com permissões personalizadas/diversas) e 2 foram classificados como *No Permissions Required* (*apps* que não precisam de permissão).

Figura 3.4 Tela Grupo de Permissões

3.4 Seleção de Aplicações Open Source

Para a análise de políticas de privacidade e permissões, a cada *release* - lançamento de uma nova versão oficial de um produto de software -, foram selecionadas 10 aplicações Android *open source* do GitHub, uma das plataformas de hospedagem de código-fonte mais utilizadas por desenvolvedores de software. São elas: AdvancedWebView, BluetoothChat, LocationAddress, LocationManager, Markhor, Ninja, Smsmms, Testdpc, Wger-android e Wifitool. Na seleção dessas, foram levados em conta *releases* e permissões, nas quais, de 194 métodos/violações, 42

foram abordadas pelas 10 aplicações.

Dessas aplicações, apenas Wger-android está na *Google Play Store* [17] e apresenta uma política de privacidade publicada [18]. A aplicação Ninja não foi colocada na loja online, porque ela ativa a execução em *background* de vídeos do YouTube, violando os termos de serviço definidos pelo YouTube API.

3.5 Extração e Análise de dados

Nesta fase, buscou-se obter as informações das ferramentas para analisá-las de acordo com os critérios previamente definidos na seção 3.3.

Para a extração dos dados dos aplicativos de um dispositivo móvel foi desenvolvida uma aplicação denominada Check Permissions, que verifica quais *apps* apresentaram variação na quantidade de permissões e quais não apresentaram, além de apontar quais não requerem qualquer tipo de permissão.

O *app* Check Permissions foi implementado utilizando a ferramenta Android Studio 3.0 na linguagem de programação Java juntamente com XML (interface gráfica das telas). Além disso, foi escolhido o padrão arquitetural MVC (Model View Controller), padrão de projeto de software que separa a interface do usuário (View) das regras de negócio e dados (Model), usando um mediador (Controller) para conectar o modelo à view. Como regra, o sistema Android já utiliza o MVC com os arquivos XML trabalhando como uma view.

Nesta análise foram avaliados três parâmetros de cada aplicativo. Sendo eles: número da versão, quantidade de permissões e data da última atualização. Vale ressaltar que, nos códigos-fonte Android, a extração dos dados ocorreu analisando manualmente cada um dos 10 repositórios *open source* selecionados.

Foi utilizado o plugin PoliDroid-AS previamente descrito neste capítulo integrado com a IDE Android Studio. Para a entrada, o mapeamento e a ontologia foram usados respectivamente os arquivos testPrivacy.txt, mapping.csv e ontology.owl.

CAPÍTULO 4

Análise dos Resultados

Neste capítulo são apresentados os resultados da análise de permissões e violações das políticas de privacidade utilizando a aplicação implementada neste trabalho (Check Permissions) e o Polidroid-AS, ambos descritos no Capítulo 3. Para o desenvolvimento do Check Permissions foi utilizado um computador com processador Intel®Core i5, com 2,3Ghz de clock e 4Gb de memória RAM.

4.1 Aplicações do Dispositivo Móvel

Em aplicativos instalados da *Google Play Store* em um dispositivo móvel, foi verificado que dos 41 aplicativos analisados com relação a quantidade de permissões, 26 deles não variaram, 13 variaram e apenas 2 não tiveram permissões requeridas, como pode ser visto na Tabela 4.1.

Tabela 4.1 Aplicativos Verificados pelo Check Permissions

Sem variações	Com variações	Sem permissões
Agenda, Ajuda do Dispositivo, Alert Me, App Box, BB, Chrome, Configurar, Contatos, Documentos, Downloads, Drive, Facebook Lite, Fotos, Google Play Filmes e TV, Google Play Música, Hangouts, Moto, Notificações Motorola, Nuo, Pluto, Radio FM, Serviços TIM, Slack, Snapchat, Telefone e Uber.	Câmera, CittaMobi, DFNDR Security, Gmail, Google, Google Play Store, Itaú, Maps, Mensagens, Nubank, Tinder, WhatsApp e YouTube.	Calculadora e Check Permissions

Para os 13 aplicativos que apresentaram variação na quantidade de permissões no momento da atualização de suas versões, foram realizadas nove coletas de dados, com início em 31/08/2017 e término em 07/12/2017. Essas alterações nas permissões de um *app* não ficam explícitas para o usuário. Ele só saberá quais permissões são requisitadas quando abre o *app*, após instalá-lo. Todavia, por meio do aplicativo Check Permissions é possível que o usuário tenha essa informação.

Na Tabela A.2 são apresentados os resultados da análise para os seguintes *apps*: Câmera, CittaMobi e DFNDR Security.

- **Câmera:** Houve apenas uma alteração no número de permissões, onde duas dessas foram adicionadas de uma vez, de 25 para 27 (06/11/2017 - 18/11/2017). As permissões adicionadas foram:
  - android.permission.ACCESS_FINE_LOCATION
  - android.permission.ACCESS_NETWORK_STATE
- **CittaMobi:** Houve duas alterações no número de permissões, de 20 para 21 (31/08/2017 - 30/09/2017) e de 21 para 22 (31/10/2017 - 06/11/2017). As permissões adicionadas foram:
  - com.google.android.c2dm.permission.RECEIVE
  - br.com.cittabus.permission.C2D_MESSAGE
- **DFNDR Security:** Houve duas alterações no número de permissões, de 57 para 58 (31/08/2017 - 30/09/2017) e de 58 para 60 (31/10/2017 - 06/11/2017). As permissões adicionadas foram:
  - com.psafe.DATAMAP_PERMISSION
  - com.psafe.crossappfeature.PERMISSION
  - com.psafe.msuite.permission.C2D_MESSAGE

Na Tabela A.3 são apresentados os resultados da análise para os seguintes *apps*: Gmail, Google e Google Play Store.

- **Gmail:** Houve apenas 1 alteração no número de permissões, de 42 para 43 (18/11/2017 - 30/11/2017), como pode ser visto na Figura 4.1. A permissão adicionada foi:
  - com.google.android.c2dm.permission.RECEIVE

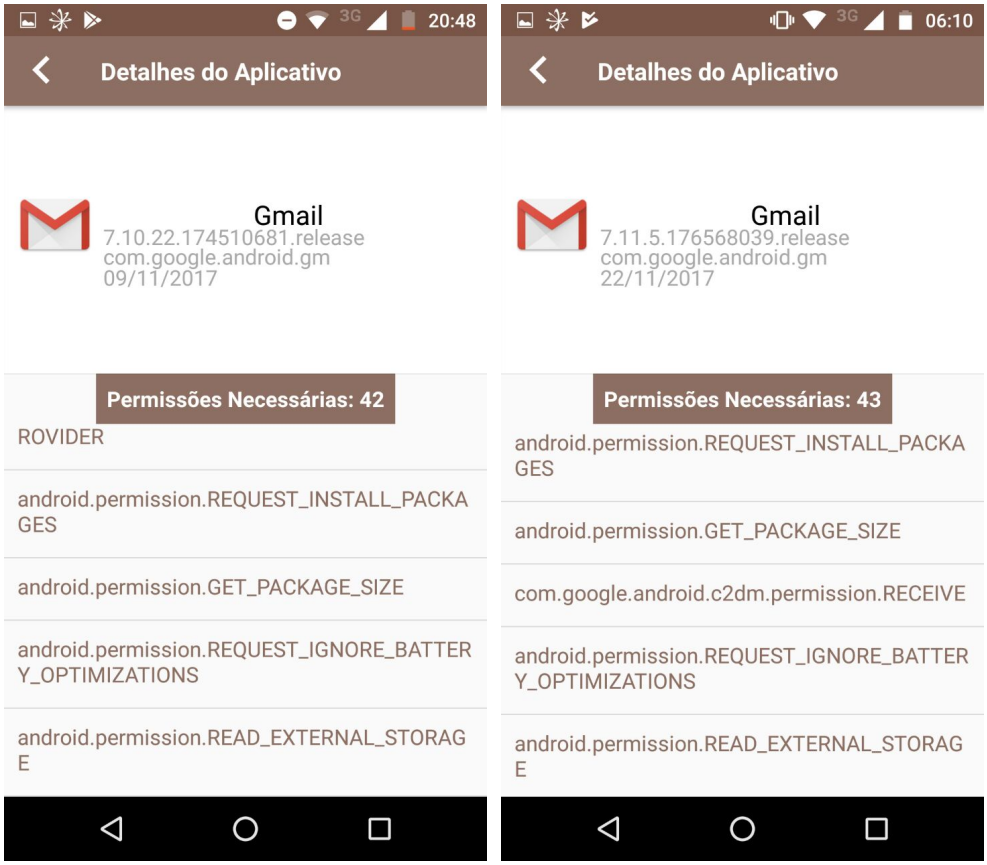


Figura 4.1 Gmail: (a) Versão 7.10.22.174510681.release; (b) Versão 7.11.5.176568039.release

- **Google:** Houve apenas 1 alteração no número de permissões, de 84 para 83 (06/11/2017 - 18/11/2017). A permissão retirada foi:
  - android.permission.SET_WALLPAPER
- **Google Play Store:** Houve três alterações no número de permissões, de 77 para 78 (30/09/2017 - 14/10/2017), de 78 para 79 (06/11/2017 - 18/11/2017) e de 79 para 80 (18/11/2017 - 30/11/2017). As permissões adicionadas foram:
  - com.google.android.c2dm.permission.RECEIVE

  - com.android.vending.billing.IN_APP_NOTIFY.permission.C2D_MESSAGE
  - com.android.vending.permission.C2D_MESSAGE

Na Tabela A.4 são apresentados os resultados da análise para os seguintes *apps*: Itaú, Maps e Mensagens.

- **Itaú:** Houve duas alterações no número de permissões, de 22 para 23 (31/08/2017 - 30/09/2017) e 23 para 25 (06/11/2017 - 18/11/2017). As permissões adicionadas foram:
  - com.itau.permission.C2D_MESSAGE
  - br.com.itau.security.WDID
  - com.itau.messenger.permission.C2D_MESSAGE
- **Maps:** Houve apenas uma alteração no número de permissões, de 33 para 34 (31/08/2017 - 30/09/2017). A permissão adicionada foi:
  - android.permission.RECEIVE_BOOT_COMPLETED
- **Mensagens:** Houve apenas uma alteração no número de permissões, de 45 para 46 (30/11/2017 - 07/12/2017). Para este *app*, os dados começaram a ser coletados a partir do dia 30/09/2017. A permissão adicionada foi:
  - com.cequint.ecid.CALLER_ID_EXTERNAL_LOOKUP_SMS

Na Tabela A.5 são apresentados os resultados da análise para os seguintes *apps*: Nubank, Tinder e WhatsApp.

- **Nubank:** Houve apenas uma alteração no número de permissões, de 17 para 18 (31/10/2017 - 06/11/2017). A permissão adicionada foi:
  - com.nu.production.permission.C2D_MESSAGE
- **Tinder:** Houve apenas uma alteração no número de permissões, de 18 para 19 (31/08/2017 - 30/09/2017), como pode ser visto na Figura 4.2. A permissão adicionada foi:
  - android.permission.RECEIVE_BOOT_COMPLETED

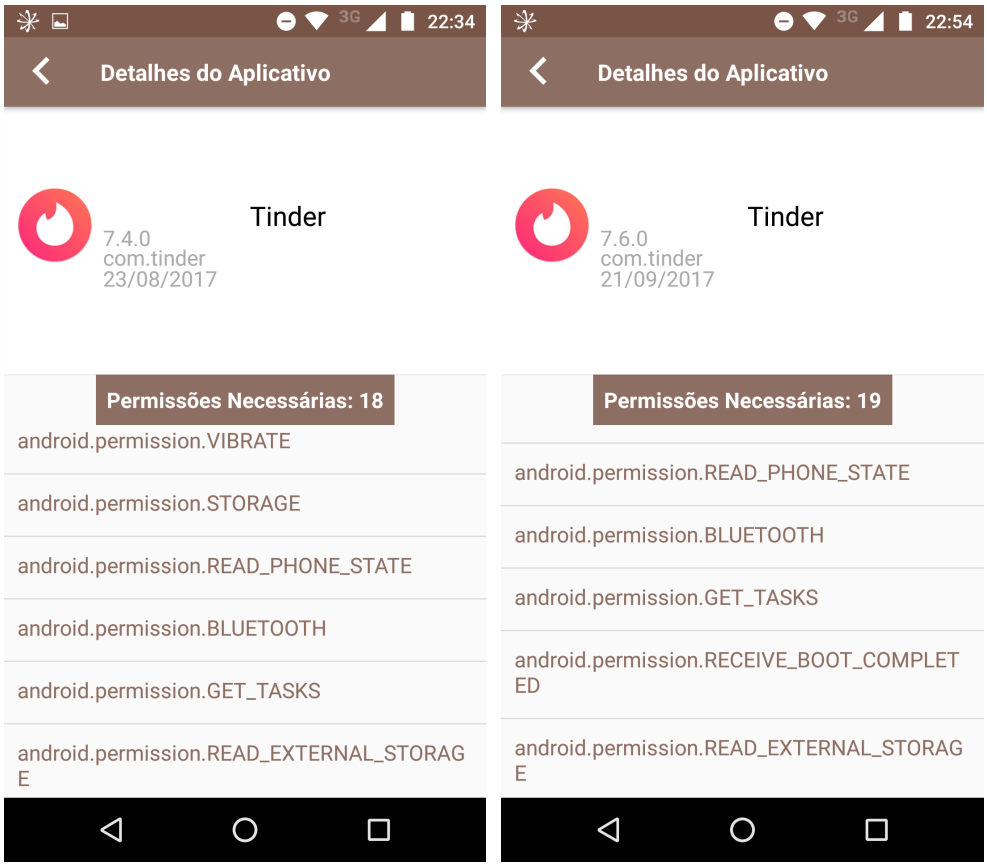


Figura 4.2 Tinder: (c) Versão 7.4.0; (d) Versão 7.6.0

- **WhatsApp:** Houve apenas uma alteração no número de permissões, de 50 para 51 (30/11/2017 - 07/12/2017), como pode ser visto na Figura 4.3. A permissão adicionada foi:
  - com.whatsapp.permission.REGISTRATION

Na Tabela A.6 são apresentados os resultados da análise para o *app* do YouTube.

- **YouTube:** Houve apenas uma alteração no número de permissões, de 34 para 33 (31/08/2017 - 30/09/2017). A permissão retirada foi:
  - com.google.android.googleapps.permission.GOOGLE_AUTH

Nas tabelas A.7 e A.8 são apresentadas as permissões e suas respectivas descrições encontradas nas 13 aplicações do dispositivo móvel.

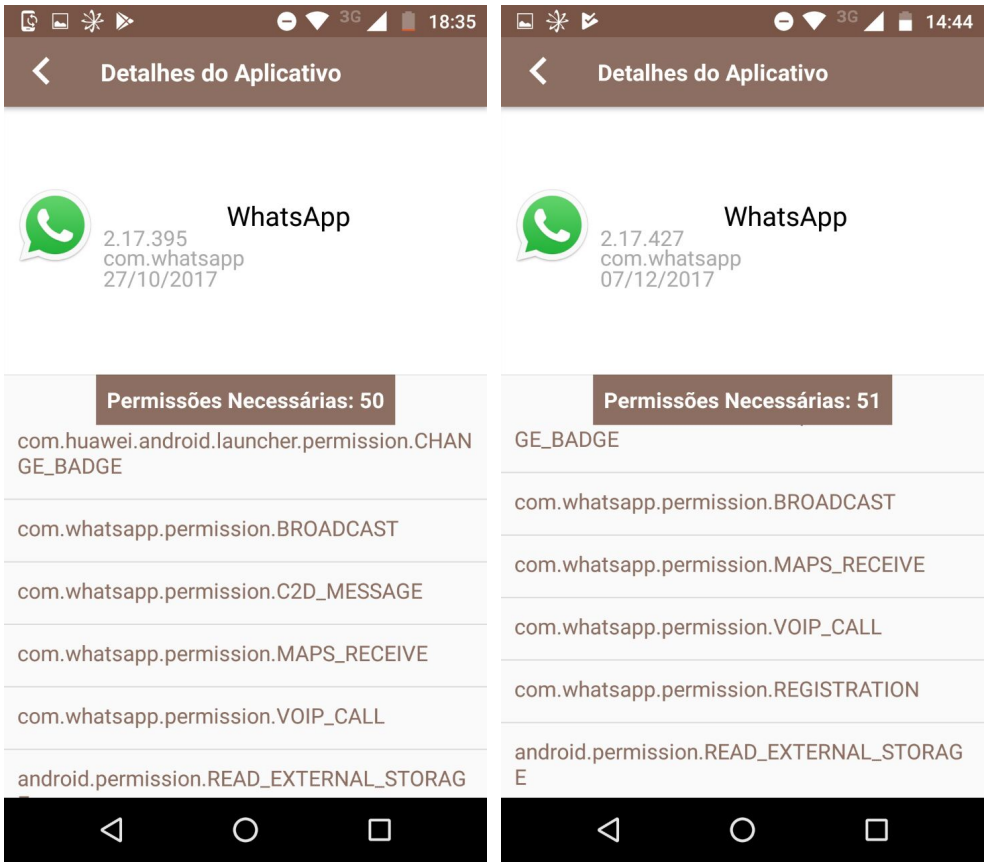


Figura 4.3 WhatsApp: (e) Versão 2.17.395; (f) Versão 2.17.427

Analisando as 13 aplicações que tiveram variação na quantidade de permissões, pode-se perceber que em duas dessas houve diminuição no número de permissões (Maps e YouTube) e nas demais onze houve aumento em pelo menos uma atualização de versão. As permissões mais adicionadas foram: android.permission.RECEIVE_BOOT_COMPLETED (permissão normal que consente a um aplicativo receber o ACTION_BOOT_COMPLETED que é transmitido após o sistema terminar a sua inicialização) e permission.C2D_MESSAGE (permissão que impede que outros *apps* se registrem e recebam as mensagens do aplicativo, trabalhando também com *push notifications*, que são mensagens de alerta enviadas ao aparelho móvel do usuário para notificá-lo na tela deste).

Isso mostra que a tendência natural é a cada atualização de versão de um *app*, a quantidade de permissões aumentar devido à novas permissões que surgem por parte da Google e permissões criadas pelas empresas dos aplicativos.

4.2 Aplicações Open Source

Na análise de detecção de violações de privacidade foram verificadas as seguintes aplicações:

- **AdvancedWebView**: Aplicação que apresenta um componente de sistema com tecnologia do Google Chrome, permitindo que aplicativos Android exibam conteúdo aprimorado da Web [19]. Possui cinco *releases* e nenhuma permissão, como pode ser visto na Tabela A.9.

- **BluetoothChat**: Aplicação que implementa um bate-papo de texto bidirecional através de bluetooth entre dois dispositivos Android, usando todos os recursos fundamentais da API Bluetooth [20]. Não possui nenhum *release* e duas permissões, como pode ser visto na Tabela A.10. Suas permissões são:

  - android.permission.BLUETOOTH_ADMIN
  - android.permission.BLUETOOTH

- **LocationAddress**: Aplicação que usa a API Geocode para exibir a localização de um dispositivo, como um endereço [21]. Não possui nenhum *release* e uma permissão, como pode ser visto na Tabela A.11. Sua única permissão é:

  - android.permission.ACCESS_FINE_LOCATION

- **LocationManager**: Aplicação que recupera a localização atual do usuário [22]. Possui cinco *releases* e quatro permissões, como pode ser visto na Tabela A.12. Suas permissões são:

  - android.permission.ACCESS_NETWORK_STATE
  - android.permission.INTERNET
  - android.permission.ACCESS_FINE_LOCATION
  - android.permission.ACCESS_COARSE_LOCATION

- **Markhor**: Aplicação que funciona como utilidades para projetos Android em geral [23]. Possui sete *releases* e nenhuma permissão, como pode ser visto na Tabela A.13.

- **Ninja**: Aplicação que funciona como um navegador web simples, que permite abrir links em segundo plano sem deixar os aplicativos favoritos, além de capturar toda a tela de uma página [24]. Possui 25 *releases* e seis permissões, como pode ser visto na Tabela A.14. Suas permissões são:
  - android.permission.INTERNET
  - android.permission.ACCESS_NETWORK_STATE
  - android.permission.ACCESS_COARSE_LOCATION
  - android.permission.ACCESS_FINE_LOCATION
  - android.permission.READ_EXTERNAL_STORAGE
  - android.permission.WRITE_EXTERNAL_STORAGE
- **Smsmms**: APIs que o Google até agora deixou de fora do ecossistema Android, com a função de enviar facilmente qualquer tipo de mensagem, procurando o código-fonte ou não [25]. Possui 4 *releases* e 15 permissões, como pode ser visto na Tabela A.15. Suas permissões são:
  - android.permission.SEND_SMS
  - android.permission.READ_SMS
  - android.permission.WRITE_SMS
  - android.permission.RECEIVE_SMS
  - android.permission.RECEIVE_MMS
  - android.permission.READ_PHONE_STATE
  - android.provider.Telephony.SMS_RECEIVED
  - android.permission.WAKE_LOCK
  - android.permission.INTERNET
  - android.permission.ACCESS_WIFI_STATE
  - android.permission.CHANGE_WIFI_STATE
  - android.permission.CHANGE_NETWORK_STATE

  - android.permission.ACCESS_NETWORK_STATE
  - android.permission.WRITE_EXTERNAL_STORAGE
  - android.permission.WRITE_SETTINGS

- **Testdpc**: Aplicação projetada para ajudar EMMs, ISVs e OEMs a testarem suas aplicações e plataformas em um perfil gerenciado do Android for Work, ou seja, perfil de trabalho. Serve como um aplicativo de teste para flexibilizar as APIs disponíveis para o Android for Work [26]. Possui 12 *releases* e 8 permissões, como pode ser visto na Tabela A.16. Suas permissões são:

  - android.permission.GET_ACCOUNTS
  - android.permission.MANAGE_ACCOUNTS
  - android.permission.PACKAGE_USAGE_STATS
  - android.permission.ACCESS_WIFI_STATE
  - android.permission.CHANGE_WIFI_STATE
  - android.permission.INTERNET
  - android.permission.RECEIVE_BOOT_COMPLETED
  - android.permission.ACCESS_NETWORK_STATE

- **Wger-android**: Aplicação livre e *open source* que funciona como um gerenciador de treinos físicos [27]. Possui três *releases* e uma permissão, como pode ser visto na Tabela A.17. Sua única permissão é:

  - android.permission.INTERNET

- **Wifitool**: É uma aplicação que ajuda a habilitar e conectar a uma rede WI-FI (internet sem fio) [28]. Possui 7 *releases* e 4 permissões, como pode ser visto na Tabela A.18. Suas permissões são:

  - android.permission.ACCESS_NETWORK_STATE
  - android.permission.ACCESS_WIFI_STATE

- android.permission.CHANGE_NETWORK_STATE
- android.permission.CHANGE_WIFI_STATE

Nas tabelas A.19, A.20 e A.21 são apresentadas as permissões e suas respectivas descrições encontradas nas 10 aplicações *open source*.

Analisando as 10 aplicações *open source* do GitHub, foi possível observar que, independentemente da variação do número de permissões ou *releases*, houve violações nas políticas de privacidade (métodos API estavam presentes no código-fonte acessando, por exemplo, sms, gps e outras informações dos usuários). Um exemplo disso é a aplicação AdvancedWebView, que não possui nenhuma permissão, mas apresenta duas potenciais violações (browser type e installed applications). A aplicação Testdpc contém 12 releases e 6 potenciais violações que são desconhecidas pelos usuários (wi-fi access point information, user information, installed applications, application software, unique application number e network measurements). Outro exemplo é a aplicação Wger-android que contém 1 release e 3 permissões, apresentando 3 potenciais violações de privacidade (contacts, url e network measurements), mesmo possuindo políticas de privacidade publicada.

Para estas aplicações *open source* do GitHub, mesmo as que possuem políticas de privacidade publicadas, foram definidas políticas de privacidade (arquivo testPrivacy.txt) utilizando o PoliDroid-AS juntamente com os arquivos mapping.csv e ontology.owl, para detectar possíveis violações em códigos-fonte. Diferentemente dos *apps* da *Google Play Store*, no qual essas políticas de privacidade são feitas por empresas de aplicativos que possuem legistas para essa tarefa ou desenvolvedores com experiência no assunto para tratar isso.

Para os usuários, fica claro que novas regras de políticas de privacidade para aplicativos Android ajudam a aumentar a confiança dos seus usuários, que passam a testar novos aplicativos e melhorar a qualidade dos *apps* produzidos pelos desenvolvedores. Porém, é necessário que especialistas tenham conhecimento sobre o código-fonte da aplicação para definir corretamente as politicas de privacidade do *app* a serem mostradas ao usuário.

CAPÍTULO 5

Conclusão e Trabalhos Futuros

Com a popularização dos smartphones, aplicativos têm sido utilizados sem que os usuários possam se dar conta de questões de segurança, como sua privacidade digital e como as permissões requeridas durante a instalação de um aplicativo.

Para entender isso, foi realizada uma análise em aplicações do Google Play Store e GitHub, a fim de identificar permissões e violações de privacidade.

Com base nos resultados obtidos, observa-se que houve variações significativas na quantidade de permissões em alguns aplicativos da *Google Play Store* a cada atualização de versão, além de muitas violações de privacidade em códigos-fonte de aplicações do *GitHub*, independentemente do número de permissões ou *releases*.

Isso mostra que o sistema de permissões no Android não garante aos usuários a segurança no controle dos seus dados, visto que na primeira vez que instala um *app* é apresentado ao usuário apenas grupos de permissões que o aplicativo precisa, não as permissões individuais. No momento que a versão de um *app* é atualizado, as modificações de permissões não são explícitas ao usuário, o que demonstra que essa administração das permissões é limitada e precisa de esclarecimentos, visto que acaba gerando muita insegurança no momento da instalação de um aplicativo.

Quanto as violações de privacidade em códigos-fonte de aplicações *open source*, a consistência do código-fonte e a política de privacidade são deixadas para desenvolvedores que têm pouca experiência ou interesse nas legalidades envolvidas, o que ficou bem visível em repositórios de aplicações Android no GitHub, que apresentaram muitas violações. O plugin PoliDroid-AS ajudou na detecção dessas violações nos referidos repositórios, uma vez que apresenta uma estrutura que aborda esse problema, detectando tais violações das políticas de privacidade no código do aplicativo Android. Além disso, seu uso descobriu em potencial 42 violações fortes e fracas de um total de 194, que foram mapeadas de 10 aplicações Android *open source*.

Algumas alternativas que podem ser aplicadas em trabalhos futuros:

- Testar a aplicação desenvolvida em vários aparelhos móveis de diferentes versões do sistema Android;
- Implementar uma ferramenta que automatize o processo de verificação de aplicações *open source* a fim de identificar violações das políticas de privacidade;
- Mapear mais métodos de API e frases que identificam políticas de privacidade estabelecidas por especialistas, como por exemplo activity recognition;
- Analisar novas permissões que a Google ou aplicativos possam estabelecer;
- Verificar como funciona o sistema de permissões de *apps* em outros sistemas operacionais, como por exemplo iOS.

Referências Bibliográficas

[1] "Número de aparelhos móveis." https://istoe.com.br/386512_NUMERO+DE+APARELHOS+MOVEIS+JA+SUPERA+O+DE+PESSOAS+NO+MUNDO/. Acessado em 22/09/2017.

[2] "Aplicativo móvel." http://blog.stone.com.br/aplicativo-movel/. Acessado em 04/09/2017.

[3] "Oha membros." http://www.openhandsetalliance.com/oha_members.html. Acessado em 10/10/2017.

[4] "Oha faq." http://www.openhandsetalliance.com/oha_faq.html. Acessado em 10/10/2017.

[5] "Android sobre." http://developer.android.com/about/index.html. Acessado em 12/10/2017.

[6] R. R. Lecheta, *Google Android-5ª Edição: Aprenda a criar aplicações para dispositivos móveis com o Android SDK*. Novatec Editora, 2015.

[7] "Permissões do sistema." https://developer.android.com/training/permissions/index.html. Acessado em 02/10/2017.

[8] "Intent." http://developer.android.com/guide/components/intents-filters.html. Acessado em 15/10/2017.

[9] R. Slavin, X. Wang, M. B. Hosseini, J. Niu, J. Bhatia, and T. D. Breaux, "Polidroid-as: A privacy policy alignment plugin for android studio,"

[10] N. Guarino, "Formal ontology, conceptual analysis and knowledge representation," *International journal of human-computer studies*, vol. 43, no. 5-6, pp. 625–640, 1995.

[11] "What is an ontology?." http://www-ksl.stanford.edu/kst/what-is-an-ontology.html. Acessado em 10/11/2017.

[12] U. Germann, "Making semantic interpretation parser-independent," in *Conference of the Association for Machine Translation in the Americas*, pp. 286–299, Springer, 1998.

[13] K. Olmstead and M. Atkinson, "Apps permissions in the google play store," 2015.

[14] J. R. Reidenberg, T. Breaux, L. F. Cranor, B. French, A. Grannis, J. T. Graves, F. Liu, A. McDonald, T. B. Norton, and R. Ramanath, "Disagreeable privacy policies: Mismatches between meaning and users' understanding," *Berkeley Tech. LJ*, vol. 30, p. 39, 2015.

[15] R. Slavin, X. Wang, M. B. Hosseini, J. Hester, R. Krishnan, J. Bhatia, T. D. Breaux, and J. Niu, "Toward a framework for detecting privacy policy violations in android application code," in *Proceedings of the 38th International Conference on Software Engineering*, pp. 25–36, ACM, 2016.

[16] M. A. Musen, "The protégé project: a look back and a look forward," *AI matters*, vol. 1, no. 4, pp. 4–12, 2015.

[17] "Wger-android google play store." https://play.google.com/store/apps/details?id=de.wger. Acessado em 01/12/2017.

[18] "Wger-android políticas de privacidade." https://wger.de/en/software/terms-of-service. Acessado em 01/12/2017.

[19] "Aplicação advancedwebview." https://github.com/delight-im/Android-AdvancedWebView/tree/master/Source. Acessado em 06/11/2017.

[20] "Aplicação bluetoothchat." https://github.com/googlesamples/android-BluetoothChat. Acessado em 30/10/2017.

[21] "Aplicação locationaddress." https://github.com/googlesamples/android-play-location/tree/master/LocationAddress. Acessado em 05/11/2017.

[22] "Aplicação locationmanager." https://github.com/yayaa/LocationManager. Acessado em 05/11/2017.

[23] "Aplicação markhor." https://github.com/shopgun/markhor. Acessado em 30/10/2017.

[24] "Aplicação ninja." https://github.com/mthli/Ninja. Acessado em 06/11/2017.

[25] "Aplicação smsmms." https://github.com/klinker41/android-smsmms. Acessado em 30/10/2017.

[26] "Aplicação testdpc." https://github.com/googlesamples/android-testdpc. Acessado em 06/11/2017.

[27] "Aplicação android." https://github.com/wger-project/android. Acessado em 18/11/2017.

[28] "Aplicação wifitool." https://github.com/yandex-qatools/android-wifitool. Acessado em 30/10/2017.

[29] "Android security." https://source.android.com/security/. Acessado em 10/09/2017.

APÊNDICE A

Resultados

Tabela A.1 Grupos de permissões e permissões perigosas

Grupo de Permissões	**Permissões Perigosas**
CALENDAR	android.permission.READ_CALENDAR android.permission.WRITE_CALENDAR
CAMERA	android.permission.CAMERA
CONTACTS	android.permission.READ_CONTACTS android.permission.WRITE_CONTACTS android.permission.GET_ACCOUNTS
LOCATION	android.permission.ACCESS_FINE_LOCATION android.permission.ACCESS_COARSE_LOCATION
MICROPHONE	android.permission.RECORD_AUDIO
PHONE	android.permission.READ_PHONE_STATE android.permission.CALL_PHONE android.permission.READ_CALL_LOG android.permission.WRITE_CALL_LOG com.android.voicemail.permission.ADD_VOICEMAIL android.permission.PROCESS_OUTGOING_CALLS android.permission.USE_SIP
SENSORS	android.permission.BODY_SENSORS
SMS	android.permission.SEND_SMS android.permission.RECEIVE_SMS android.permission.READ_SMS android.permission.RECEIVE_WAP_PUSH android.permission.RECEIVE_MMS
STORAGE	android.permission.READ_PHONE_STATE android.permission.WRITE_PHONE_STATE

Tabela A.2 Aplicativos com variações - Parte 1

Aplicativo	**Data de Verificação**	**Versão**	**Quantidade de Permissões**	**Última Atualização**
Câmera	31/08/2017	6.0.43.10	25	07/09/2016
	30/09/2017	6.0.43.10	25	07/09/2016
	14/10/2017	6.0.43.10	25	07/09/2016
	22/10/2017	6.0.43.10	25	07/09/2016
	31/10/2017	6.0.43.10	25	07/09/2016
	06/11/2017	6.0.43.10	25	07/09/2016
	18/11/2017	6.0.86.7	27	08/11/2017
	30/11/2017	6.0.86.7	27	08/11/2017
	07/12/2017	6.0.86.7	27	08/11/2017
CittaMobi	31/08/2017	5.03	20	24/08/2017
	30/09/2017	5.1.2	21	29/09/2017
	14/10/2017	5.2.0	21	05/10/2017
	22/10/2017	5.2.0	21	05/10/2017
	31/10/2017	5.2.2	21	30/10/2017
	06/11/2017	5.3.0	22	05/11/2017
	18/11/2017	5.3.7	22	17/11/2017
	30/11/2017	5.4.0	22	28/11/2017
	07/12/2017	5.4.0	22	28/11/2017
DFNDR Security	31/08/2017	4.1.6	57	20/08/2017
	30/09/2017	4.1.10	58	27/09/2017
	14/10/2017	5.0.7	58	10/10/2017
	22/10/2017	5.0.7	58	10/10/2017
	31/10/2017	5.0.7	58	10/10/2017
	06/11/2017	5.0.12	60	01/11/2017
	18/11/2017	5.0.12	60	01/11/2017
	30/11/2017	5.1.1	60	21/11/2017
	07/12/2017	5.1.1	60	21/11/2017

Tabela A.3 Aplicativos com variações - Parte 2

Aplicativo	Data de Verificação	Versão	Quantidade de Permissões	Última Atualização
Gmail	31/08/2017	7.8.13.166937981.release	42	30/08/2017
	30/09/2017	7.8.27.168289052.release	42	15/09/2017
	14/10/2017	7.9.10.169126262.release	42	03/10/2017
	22/10/2017	7.9.24.172525262.release	42	20/10/2017
	31/10/2017	7.10.8.172533986.release	42	31/10/2017
	06/11/2017	7.10.8.172533986.release	42	31/10/2017
	18/11/2017	7.10.22.174510681.release	42	09/11/2017
	30/11/2017	7.11.5.176568039.release	43	22/11/2017
	07/12/2017	7.11.5.177402951.release	43	01/12/2017
Google	31/08/2017	7.8.22.21.arm	84	13/08/2017
	30/09/2017	7.11.24.21.arm	84	25/09/2017
	14/10/2017	7.11.24.21.arm	84	06/10/2017
	22/10/2017	7.13.28.21.arm	84	20/10/2017
	31/10/2017	7.13.28.21.arm	84	20/10/2017
	06/11/2017	7.14.21.21.arm	83	03/11/2017
	18/11/2017	7.15.22.21.arm	83	17/11/2017
	30/11/2017	7.15.22.21.arm	83	17/11/2017
	07/12/2017	7.16.19.21.arm	83	07/12/2017
Google Play Store	31/08/2017	8.1.31.S-all [0] [PR] 165266818	77	25/08/2017
	30/09/2017	8.2.38.T-all [0] [FP] 169346653	77	25/09/2017
	14/10/2017	8.3.41.U-all [0] [FP] 170066753	78	06/10/2017
	22/10/2017	8.3.41.U-all [0] [FP] 170066753	78	06/10/2017
	31/10/2017	8.3.41.U-all[0] [FP] 170066753	78	06/10/2017
	06/11/2017	8.3.41.U-all[0] [FP] 170066753	78	06/10/2017
	18/11/2017	8.3.73.U-all [0] [FP] 173262113	79	06/11/2017
	30/11/2017	8.4.19.V-all [0] [FP] 175058788	80	20/11/2017
	07/12/2017	8.4.19.V-all [0] [FP] 175058788	80	20/11/2017

Tabela A.4 Aplicativos com variações - Parte 3

Aplicativo	Data de Verificação	Versão	Quantidade de Permissões	Última Atualização
Itaú	31/08/2017	6.2.0	22	25/08/2017
	30/09/2017	6.2.5	23	22/09/2017
	14/10/2017	6.2.5	23	22/09/2017
	22/10/2017	6.2.6	23	20/10/2017
	31/10/2017	6.2.6	23	20/10/2017
	06/11/2017	6.2.6	23	20/10/2017
	18/11/2017	6.3.0	25	17/11/2017
	30/11/2017	6.3.0	25	17/11/2017
	07/12/2017	6.3.0	25	17/11/2017
Maps	31/08/2017	9.60.1	33	31/08/2017
	30/09/2017	9.62.1	34	20/09/2017
	14/10/2017	9.63.1	34	06/10/2017
	22/10/2017	9.64.1	34	21/10/2017
	31/10/2017	9.64.1	34	21/10/2017
	06/11/2017	9.65.1	34	03/11/2017
	18/11/2017	9.66.1	34	15/11/2017
	30/11/2017	9.66.1	34	15/11/2017
	07/12/2017	9.67.1	34	01/12/2017
Mensagens	31/08/2017	-	-	-
	30/09/2017	2.5.211	45	29/09/2017
	14/10/2017	2.5.212	45	06/10/2017
	22/10/2017	2.5.212	45	06/10/2017
	31/10/2017	2.5.212	45	06/10/2017
	06/11/2017	2.5.212	45	06/10/2017
	18/11/2017	2.5.212	45	06/10/2017
	30/11/2017	2.5.212	45	06/10/2017
	07/12/2017	2.7.030	46	01/12/2017

Tabela A.5 Aplicativos com variações - Parte 4

Aplicativo	Data de Verificação	Versão	Quantidade de Permissões	Última Atualização
Nubank	31/08/2017	4.27.0-minApi21	17	26/08/2017
	30/09/2017	4.28.0-minApi21	17	28/09/2017
	14/10/2017	4.28.0-minApi21	17	28/09/2017
	22/10/2017	4.29.4-minApi21	17	22/10/2017
	31/10/2017	4.29.4-minApi21	17	22/10/2017
	06/11/2017	4.30.0-minApi21	18	02/11/2017
	18/11/2017	4.30.1-minApi21	18	16/11/2017
	30/11/2017	4.30.2-minApi21	18	28/11/2017
	07/12/2017	4.30.2-minApi21	18	07/12/2017
Tinder	31/08/2017	7.4.0	18	23/08/2017
	30/09/2017	7.6.0	19	21/09/2017
	14/10/2017	8.0.0	19	21/09/2017
	22/10/2017	8.0.1	19	15/10/2017
	31/10/2017	8.0.1	19	15/10/2017
	06/11/2017	8.1.0	19	04/11/2017
	18/11/2017	8.1.1	19	11/11/2017
	30/11/2017	8.2.0	19	30/11/2017
	07/12/2017	8.2.1	19	05/12/2017
WhatsApp	31/08/2017	2.17.296	50	13/08/2017
	30/09/2017	2.17.323	50	08/09/2017
	14/10/2017	2.17.351	50	03/10/2017
	22/10/2017	2.17.351	50	03/10/2017
	31/10/2017	2.17.395	50	27/10/2017
	06/11/2017	2.17.395	50	27/10/2017
	18/11/2017	2.17.395	50	27/10/2017
	30/11/2017	2.17.395	50	27/10/2017
	07/12/2017	2.17.427	51	07/12/2017

Tabela A.6 Aplicativos com variações - Parte 5

Aplicativo	**Data de Verificação**	**Versão**	**Quantidade de Permissões**	**Última Atualização**
YouTube	31/08/2017	12.32.60	34	31/08/2017
	30/09/2017	12.36.56	33	22/09/2017
	14/10/2017	12.39.60	33	11/10/2017
	22/10/2017	12.40.60	33	19/10/2017
	31/10/2017	12.42.59	33	27/10/2017
	06/11/2017	12.42.59	33	27/10/2017
	18/11/2017	12.44.53	33	17/11/2017
	30/11/2017	12.45.56	33	30/11/2017
	07/12/2017	12.45.56	33	30/11/2017

Tabela A.7 Permissões das Aplicações de Dispositivos Móveis - Parte 1

Permissões	Descrição	Aplicações
android.permission. ACCESS_FINE_LOCATION	Permissão perigosa para *apps* acessarem uma localização específica	Câmera
android.permission. ACCESS_NETWORK_STATE	Permissão normal para *apps* acessarem dados sobre redes	Câmera
com.google.android.c2dm. permission.RECEIVE	Permissão que possibilita o *app* trabalhar com *push notifications*	CittaMobi Gmail Google Play Store
br.com.cittabus.permission. C2D_MESSAGE	Permissão que impede que outros *apps* se registrem e recebam as mensagens do aplicativo	CittaMobi
android.permission. SET_WALLPAPER	Permissão normal para *apps* definirem o papel de parede	Google
com.android.vending.billing. IN_APP_NOTIFY.permission. C2D_MESSAGE	Permissão que impede que outros *apps* se registrem e recebam as mensagens do aplicativo	Google Play Store
com.android.vending.permission. C2D_MESSAGE	Permissão que impede que outros *apps* se registrem e recebam as mensagens do aplicativo	Google Play Store
com.itau.permission. C2D_MESSAGE	Permissão que impede que outros *apps* se registrem e recebam as mensagens do aplicativo	Itaú
br.com.itau.security.WDID	Permissão específica do Banco Itaú para segurança dos usuários	Itaú
com.itau.messenger.permission. C2D_MESSAGE	Permissão que impede que outros *apps* se registrem e recebam as mensagens do aplicativo	Itaú
android.permission. RECEIVE_BOOT_COMPLETED	Permissão normal para *apps* receberem o ACTION_BOOT_COMPLETED, que é transmitido após o sistema terminar a inicialização	Maps Tinder

Tabela A.8 Permissões das Aplicações de Dispositivos Móveis - Parte 2

Permissões	Descrição	Aplicações
com.cequint.ecid. CALLER_ID_EXTERNAL _LOOKUP_SMS	Permissão normal do aplicativo Mensagens em dispositivos da Motorola	Mensagens
com.nu.production.permission. C2D_MESSAGE	Permissão que impede que outros *apps* se registrem e recebam as mensagens do aplicativo	Nubank
com.psafe. DATAMAP_PERMISSION	Permissão responsável pelo mapeamento de informações de outros *apps*	DFNDR Security
com.psafe.crossappfeature. PERMISSION	Permissão que pode receber *cloud messages*	DFNDR Security
com.psafe.msuite.permission. C2D_MESSAGE	Permissão que impede que outros *apps* se registrem e recebam as mensagens do aplicativo	DFNDR Security
com.whatsapp.permission. REGISTRATION	Permissão específica para registro e identifcação de contas dos usuários	WhatsApp
com.google.android.googleapps. permission.GOOGLE_AUTH	Permissão para os *apps* verem os nomes de usuários (endereços de e-mail) da(s) conta(s) configurada(s) do Google	YouTube

Tabela A.9 Aplicação Open Source AdvancedWebView

AdvancedWebView (0 Permissões, 5 Releases)	
Método	**Potenciais Violações**
android.webkit.WebSettings.getUserAgentString()	browser type
android.content.pm.PackageManager. getInstalledApplications(int flags)	installed applications

Tabela A.10 Aplicação Open Source BluetoothChat

BluetoothChat (2 Permissões, 0 Releases)	
Método	**Potenciais Violações**
android.bluetooth.bluetoothadapter.isEnabled()	bluetooth settings
android.bluetooth.BluetoothDevice.getName()	bluetooth settings
android.bluetooth.BluetoothAdapter.getBondedDevices()	bluetooth settings

Tabela A.11 Aplicação Open Source LocationAddress

LocationAddress (1 Permissão, 0 Releases)	
Método	**Potenciais Violações**
android.location.Location.getLongitude()	longitude
android.location.Location.getLatitude()	latitude
android.location.Address.getAddressLine(int index)	location tags zip code

Tabela A.12 Aplicação Open Source LocationManager

LocationManager (4 Permissões, 5 Releases)	
Método	**Potenciais Violações**
android.location.Location.getLatitude()	latitude
android.location.Location.getLongitude()	longitude
android.location.LocationManager. requestLocationUpdates (String provider, long minTime, float minDistance, LocationListener listener)	location
android.location.LocationManager. getLastKnownLocation(String provider)	real-time location-based information
android.location.Location.getAccuracy()	gps information
android.location.Location.getTime()	time

Tabela A.13 Aplicação Open Source Markhor

Markhor (0 Permissões, 7 Releases)	
Método	**Potenciais Violações**
android.content.pm.PackageManager. getApplicationInfo(String packageName, int flags)	application software unique application number

Tabela A.14 Aplicação Open Source Ninja

Ninja (6 Permissões, 25 Releases)	
Método	**Potenciais Violações**
android.content.Context.openFileInput(String name)	files
android.webkit.WebView.getOriginalUrl()	referral url url
android.webkit.WebSettings.getUserAgentString()	browser type

Tabela A.15 Aplicação Open Source Smsmms

Smsmms (15 Permissões, 4 Releases)	
Método	**Potenciais Violações**
android.telephony.SmsMessage.getMessageBody()	chat text sms messages text messages number
android.location.Location.getLatitude()	latitude
android.location.Address.getAddressLine(int index)	location tags zip code

Tabela A.16 Aplicação Open Source Testdpc

Testdpc (8 Permissões, 12 Releases)	
Método	**Potenciais Violações**
android.net.wifi.WifiManager.getConfiguredNetworks()	wi-fi access point information
android.app.admin.DevicePolicyManager.getActiveAdmins()	user information
android.content.pm.PackageManager. getInstalledApplications(int flags)	installed applications
android.content.pm.PackageManager. getPackagesForUid(int uid)	installed applications
android.content.pm.PackageManager. getApplicationInfo(String packageName, int flags)	application software unique application number
android.net.ConnectivityManager.getActiveNetworkInfo()	network measurements

Tabela A.17 Aplicação Open Source Wger-android

Wger-android (1 Permissão, 3 Releases)	
Método	**Potenciais Violações**
android.net.MailTo.getCc()	contacts url
android.net.MailTo.getTo()	contacts url
android.net.MailTo.getBody()	url
android.webkit.CookieManager.getCookie(String url)	network measurements

Tabela A.18 Aplicação Open Source Wifitool

Wifitool (4 Permissões, 7 Releases)	
Método	**Potenciais Violações**
android.net.wifi.WifiManager.getConfiguredNetworks()	wi-fi access point information
android.net.wifi.WifiInfo.getNetworkId()	wi-fi access point information
android.net.wifi.WifiManager.getConnectionInfo()	domain servers ip address network measurements wi-fi access point information wifi signal strength
android.net.NetworkInfo.getType()	network protocol
android.net.wifi.WifiManager.getWifiState()	wi-fi access point information
android.net.ConnectivityManager.getActiveNetworkInfo()	network measurements
android.net.wifi.WifiInfo.getSSID()	wi-fi access point information

Tabela A.19 Permissões das Aplicações Open Source - Parte 1

Permissões	Descrição	Aplicações
android.permission. INTERNET	Permissão normal para *apps* abrirem sockets de rede	Smsmms Ninja LocationManager Testdpc Wger-android
android.permission. BLUETOOTH_ADMIN	Permissão normal para *apps* descobrirem e emparelharem dispositivos bluetooth.	BluetoothChat
android.permission. BLUETOOTH	Permissão normal para *apps* se conectarem a dispositivos bluetooth emparelhados	BluetoothChat
android.permission. ACCESS_FINE_LOCATION	Permissão perigosa para *apps* acessarem localização precisa	LocationAddress LocationManager Ninja
android.permission. ACCESS_NETWORK_STATE	Permissão normal para *apps* acessarem dados sobre redes	LocationManager Ninja Smsmms Testdpc Wifitool
android.permission. ACCESS_COARSE.LOCATION	Permissão perigosa para *apps* acessarem localização aproximada	LocationManager Ninja
android.permission. READ_EXTERNAL_STORAGE	Permissão perigosa para *apps* serem lidas a partir de um armazenamento externo	Ninja
android.permission. WRITE_EXTERNAL_STORAGE	Permissão perigosa para *apps* serem escritas a partir de um armazenamento externo	Ninja Smsmms
android.permission. SEND_SMS	Permissão perigosa para *apps* enviarem mensagens SMS	Smsmms

Tabela A.20 Permissões das Aplicações Open Source - Parte 2

Permissões	Descrição	Aplicações
android.permission. READ_SMS	Permissão perigosa para *apps* lerem mensagens SMS	Smsmms
android.permission. WRITE_SMS	Permissão perigosa para *apps* escreverem mensagens SMS	Smsmms
android.permission. RECEIVE_SMS	Permissão perigosa para *apps* receberem mensagens SMS	Smsmms
android.permission. RECEIVE_MMS	Permissão perigosa para *apps* monitorarem as mensagens MMS recebidas	Smsmms
android.permission. READ_PHONE_STATE	Permissão perigosa para o acesso somente de leitura ao estado do telefone, incluindo o número de telefone do dispositivo, as informações atuais da rede celular, o status de todas as chamadas em andamento e uma lista de todas as contas de telefone registradas no dispositivo	Smsmms
android.permission. WAKE_LOCK	Permissão normal para o uso do *PowerManager WakeLocks* a fim de evitar que o processador durma ou a tela escureça.	Smsmms
android.permission. ACCESS_WIFI_STATE	Permissão normal para *apps* acessarem informações sobre redes WI-FI	Smsmms Testdpc Wifitool
android.permission. CHANGE_WIFI_STATE	Permissão normal para *apps* alterarem o estado de conectividade Wi-Fi	Smsmms Testdpc Wifitool
android.permission. CHANGE_NETWORK_STATE	Permissão normal para *apps* alterarem o estado da conectividade de rede	Smsmms Wifitool

Tabela A.21 Permissões das Aplicações Open Source - Parte 3

Permissões	**Descrição**	**Aplicações**
android.permission. WRITE_SETTINGS	Permissão especial para *apps* lerem ou escreverem as configurações do sistema.	Smsmms
android.permission. GET_ACCOUNTS	Permissão especial para *apps* acessarem a lista de contas no Serviço de Contas.	Testdpc
android.permission. MANAGE_ACCOUNTS	Permissão especial para o gerenciamento de contas no Serviço de Contas	Testdpc
android.permission. PACKAGE_USAGE_STATS	Permissão para *apps* coletarem estatísticas de uso de componentes	Testdpc
android.permission. RECEIVE_BOOT_COMPLETED	Permissão normal para *apps* receberem o ACTION_BOOT_COMPLETED, que é transmitido após o sistema terminar a inicialização	Testdpc