UNIVERSIDADE FEDERAL DO PAMPA

WILSON JACOBSEN

UMA SOLUÇÃO PARA AVALIAÇÃO DE RISCOS EM APLICATIVOS PARA DISPOSITIVOS MÓVEIS

Bagé
2014

WILSON JACOBSEN

UMA SOLUÇÃO PARA AVALIAÇÃO DE RISCOS EM APLICATIVOS PARA DISPOSITIVOS MÓVEIS

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Computação da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Computação.

Orientador: Érico Marcelo Hoff do Amaral

Bagé
2014

WILSON JACOBSEN

UMA SOLUÇÃO PARA AVALIAÇÃO DE RISCOS EM APLICATIVOS PARA DISPOSITIVOS MÓVEIS

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Computação da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Computação.

Trabalho de Conclusão de Curso defendido em: 31 de Janeiro de 2015.

Banca examinadora:

_____

Prof. MSc. Érico Marcelo Hoff do Amaral
Orientador
UNIPAMPA

_____

Prof. Dr. Milton Roberto Heinen
UNIPAMPA

_____

Prof. MSc. Fábio Luís Livi Ramos
UNIPAMPA

Dedico este trabalho primeiramente a Deus, por ser essencial em minha vida, aos meus pais, minha irmã, minha namorada, e ainda aos meus professores e colegas que me auxiliaram nessa jornada.

AGRADECIMENTO

A Deus por ter me dado saúde e força para superar as dificuldades.

A minha mãe, heroína que me deu apoio, incentivo nas horas difíceis, de desânimo e cansaço.

Ao meu pai, meu ídolo que apesar de todas as dificuldades me apoiou fortaleceu e que para mim foi muito importante.

A minha irmã, que nos momentos de minha ausência dedicados ao estudo superior, sempre fiz entender que o futuro é feito a partir da constante dedicação no presente!

A minha namorada, pelo apoio e incentivo dado ao longo dessa jornada.

Ao meu professor orientador, pelo empenho dedicado à elaboração deste trabalho.

Agradeço a esta universidade e todos os professores por me proporcionarem o conhecimento não apenas racional, mas a manifestação do caráter e afetividade da educação no processo de formação profissional, por tanto que se dedicaram a mim, não somente por terem me ensinado, mas por terem me feito aprender. A palavra mestre, nunca fará justiça aos professores dedicados aos quais sem nominar terão os meus eternos agradecimentos.

A todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

“Seu trabalho vai ocupar uma grande parte da sua vida, e a única maneira de estar verdadeiramente satisfeito é fazendo aquilo que você acredita ser um ótimo trabalho. E a única maneira de fazer um ótimo trabalho é fazendo o que você ama fazer.”

Steve Jobs

RESUMO

Acompanhando a evolução tecnológica dos sistemas móveis, um conjunto de facilidades e recursos computacionais disponibilizados em larga escala aos usuários, que atualmente podem utilizar seus dispositivos para o armazenamento de dados, como recurso para localização global, acesso a redes sociais e inúmeras outras aplicações. Contudo, estas demandas devem ser calcadas sobre uma premissa básica: o uso de uma plataforma segura, a qual viabilize a redução dos riscos na perda da integridade, disponibilidade ou confiabilidade das informações de seus usuários. Nos dias de hoje, a ocorrência de um incidente de segurança, sobre um dispositivo móvel *smartphone*, *tablet* ou celular, pode ser agravada devido à grande conectividade e processamento oferecido por estes aparelhos. A restauração ou manutenção dos dados comprometidos, após uma falha de segurança, é um procedimento caro e, muitas vezes, não traz o benefício esperado. Uma alternativa para a redução deste tipo de transtorno está na adoção de ações proativas, visando a manutenção e implementação de métodos que aumentem os níveis de segurança em sistemas móveis. Como uma maneira de contenção a problemas de segurança, encadeados por tentativas de acessos não autorizados e uso indiscriminado de Apps, este trabalho propõe o estudo e criação de uma metodologia para a análise de potenciais riscos em aplicativos desenvolvidos exclusivamente para o Android, a qual é considerada a principal plataforma para dispositivos móveis disponíveis no mercado. Como resultado, busca-se a disponibilidade de uma solução para identificar o risco agregado a instalação ou manutenção de determinados aplicativos instalados em seu dispositivo móvel. Busca-se também identificar intenções maliciosas destes softwares, como a exigência de permissões abusivas, não necessárias para o funcionamento da aplicação. Este trabalho possui resultados, dentre os quais citam-se: um estudo detalhado das permissões de padrões para aplicativos Android e uma avaliação dos potenciais riscos agregados a cada uma destas, a criação e validação de um novo modelo de análise de risco, implementação de um aplicativo para automatizar o modelo criado.

Palavras-Chave: Dispositivos móveis, *smartphones*, aplicativos, segurança.

ABSTRACT

Following the technological evolution of mobile systems, a set of facilities and computing resources available on a large scale users, who can now use their devices for data storage as a resource for global location, access to social networks and many other applications. However, these demands should be justified based on a basic premise: the use of a secure platform, which makes possible the reduction of risks in the loss of integrity, availability or reliability of the information of its users. These days, the occurrence of a security incident on a smartphone mobile, tablet or mobile phone, can be aggravated by the great connectivity and processing offered by these devices. The restoration or maintenance of compromised data, after a security flaw, it is an expensive procedure and often does not bring the expected benefit. An alternative to reducing this type of disorder is the adoption of proactive measures, aimed at maintaining and implementing methods to increase safety levels in mobile systems. As a way to contain the security problems, linked by attempts of unauthorized access and indiscriminate use of Apps, this paper proposes the study and creation of a methodology for hazard analysis in applications developed exclusively for Android, which is considered the main platform for mobile devices available on the market. As a result, the availability we seek a solution to identify the aggregate risk the installation or maintenance of certain applications installed on your mobile device. The aim is to also identify malicious intent of such *software*, such as the requirement of abusive permissions, not necessary for the operation of the application. This work has results, among which we mention: a detailed study of patterns permissions for Android applications and an assessment of potential aggregate risks to each of these, the creation and validation of a new risk analysis model and implementation of an application to automate the model created.

Key words: mobile devices, smartphones, applications, security.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AMPS - *Advanced Mobile Phone System*

TDMA - *Time Division Multiple Access*

CDMA -*Code Division Multiple Access*

GSM - *Groupe Special Mobile*

PDA - *Groupe Special Mobile*

MMS - *Multimidia Messaging Service*

SMS - *Short Message Service*

GPS - *Global Positioning System*

SRAM - *Static Random Access Memory*

SDR - *Special Drawing Right*

DDR - *Double Data Rate*

SDRAM - *Synchronous dynamic random access memory*

SDK - *Android Software Development Kit*

APK - *Android Package*

ID – Identificador

HD- High *Definition*

API - Application Programming Interface

XML - Xtensible Markup Language

DDMS - *Dalvik Debug Server Monitor*

HTTP - *Hypertext Transfer Protocol*

SUMÁRIO

1 INTRODUÇÃO

A evolução da tecnologia para sistemas móveis, aliada ao crescimento do uso de sistemas de armazenamento de dados, sistemas de localização, redes sociais e também pela disponibilidade insaciável de novos aplicativos no mercado (TONIN, 2012), aumenta de forma exponencial a dependência de um dispositivo móvel confiável e seguro.

A ocorrência do incidente de segurança em celulares ou *tablets* é agravada devido à grande conectividade e processamento oferecido por estes aparelhos, e a restauração e manutenção dos dados comprometidos em um incidente são procedimentos caros e muitas vezes não trazem o benefício esperado. Uma alternativa para a redução deste tipo de transtorno está colocada em ações proativas, com o investimento na análise e correção das vulnerabilidades dos sistemas.

Um dos grandes problemas destes analisadores são as inconsistências dos dados apresentados. É perigoso ter a impressão de que não existem vulnerabilidades quando, na realidade, as brechas estão por todo lado e disponíveis aos fraudadores interessados. Por essa razão a segurança precisa envolver tecnologias, processos e pessoas num trabalho que deve ser cíclico, contínuo e persistente, com a consciência de que os resultados estarão consolidados em médio prazo.

Muniz *et al.* (2013) define a vulnerabilidade como uma fragilidade que poderia ser explorada por uma ameaça para concretizar um ataque. Em um sistema operacional ou em aplicativos existem diversas vulnerabilidades que podem ser exploradas por um *cracker*, cabe ao usuário garantir que as vulnerabilidades existentes não sejam um risco para os seus dados. Para isso devemos conhecer e corrigir as vulnerabilidades. Em um cenário empresarial ou, até mesmo, em um desktop de um usuário comum, esse problema poderia ser corrigido com scanners de vulnerabilidade como *Nessus*[1] *e Openvas*[2] (MUNIZ *et al.*, 2013), mas tal prática requer conhecimento específico e gastos que um usuário comum de *smartphone* ou *tablet* não estaria disposto a pagar. Tendo em vista essa situação, o estudo de

[1] *Software pago* de *scanner* de vulnerabilidade (Muniz, 2014).
[2] *Software free* de *scanner* de vulnerabilidade (Muniz, 2014).

novas técnicas de segurança torna-se extremamente necessário, priorizando uma tentativa de contenção a invasões e mau uso de aplicativos, logo, este trabalho propõe um estudo e criação de uma ferramenta de análise de potenciais riscos em aplicativos, objetivando fornecer uma maneira do próprio usuário saber o risco que está correndo em manter um determinado aplicativo instalado em seu dispositivo móvel.

1.1 Problema de Pesquisa

De acordo com Braga *et al.* (2012), a partir do ano de 2011 o crescimento do número de aplicativos e usuários de dispositivos móveis influenciou o surgimento de muitos aplicativos maliciosos, em particular na plataforma Android[3], a qual não se obteve uma reposta rápida dos fornecedores de produtos de segurança da informação. Segundo Enck *et. al.* (2011), a falta de resposta dos fornecedores se deu pela ausência de interesse dos usuários com a segurança e tecnologias robustas para suportar as soluções desta questão.

O uso crescente de dispositivos móveis é um exemplo contrário dos computadores, que foram criados para um ambiente corporativo de grandes empresas. Os dispositivos móveis surgiram primeiramente voltados para usuários finais e somente depois para um segmento corporativo, normalmente, influenciados pelo uso dos funcionários e pelas facilidades de mobilidade que nele são empregados. A preocupação das empresas com a segurança da nova ferramenta de trabalho acabou alavancando o mercado de segurança de dispositivos móveis e a busca de soluções eficientes e atrativas para os usuários deste cenário (BRAGA *et al.,*2012).

1.2 Objetivos

Este trabalho vislumbra implementar e validar um modelo para avaliação de potenciais riscos e vulnerabilidades encontradas em aplicativos para dispositivos móveis, a fim de garantir a manutenção dos princípios básicos da segurança da informação. Assim, existirá uma maneira para que os usuários avaliem os riscos ao

[3] **Android** é um sistema operacional baseado no núcleo do Linux para dispositivos móveis, desenvolvido pela Open Handset Alliance, liderada pela Google Inc (ANDROID, 2014).

utilizarem seu dispositivo móvel e posteriormente, irão poder escolher entre os aplicativos, aqueles que forem mais seguros.

Objetivos Específicos:

- Realizar um estudo sobre plataformas e sistemas operacionais comumente utilizados em dispositivos móveis;
- Identificar e analisar o funcionamento destes sistemas, com o foco em características, como: instalação, funcionamento, desempenho, utilização de recursos, sincronização de dados entre outros;
- Identificar e analisar os padrões adotados por estes sistemas para a definição de permissões e acesso as informações do usuário;
- Identificar e avaliar potenciais riscos e vulnerabilidades que possam causar prejuízos a privacidade dos usuários de dispositivos móveis;
- Propor um Modelo para análise e avaliação de riscos e vulnerabilidades em sistemas de dispositivos móveis;
- Implementar o modelo criado para executar nos dispositivos móveis.
- Realizar experimentos de validação da aplicação implementada.

1.3 Hipóteses

O trabalho consiste na criação e validação de um modelo de análise de risco em aplicativos para a plataforma Android, o trabalho pode apresentar as seguintes hipóteses:

H0: É possível propor e implementar uma aplicação para smartphones baseada em um modelo eficiente para a análise do risco na utilização de aplicativos em dispositivos móveis com Android.

H1: Não é possível propor ou implementar uma aplicação para análise de risco em dispositivos móveis.

2 CONCEITOS GERAIS E REVISÃO DE LITERATURA

O avanço da tecnologia para dispositivos móveis é um grande marco do século XXI. Quando os celulares foram lançados no mercado, seu tamanho era proporcionalmente grande em comparação com os atuais, e apenas pessoas com boas condições financeiras poderiam possuir um desses aparelhos, por exemplo, o modelo Dynatac 8000X, testado na década de 70, pesava mais de meio quilo e custava o equivalente a US$ 3.995 na época, o que hoje corresponderia a US$ 10 mil. (O GLOBO, 2014).

Com o tempo os dispositivos móveis passaram por uma significativa evolução, como podemos observar na Figura 1, não evoluíram somente no tamanho, mas também em protocolos de comunicação sem fio, sensores, *hardware* e funcionalidades que se aproximam dos computadores.

Figura 1: Ilustra a evolução dos celulares de 1956 à 2014.

Fonte: (Próprio Autor,2014).

A memória, o processamento e os sensores, que estão inclusos no hardware do dispositivo móvel, são utilizados por aplicativos a fim de executarem as tarefas as

quais foram criados para fazer, mas também podem ser utilizados por tarefas maliciosas que causam danos ao usuário, estes programas não são fáceis de identificar. Uma solução é a criação de ferramentas que auxiliem o usuário no processo de identificação de possíveis *malwares*. Para tanto, é importante entender alguns componentes básicos dos sistemas móveis que se encontram nessa seção. Serão apresentados e discutidos a seguir os seguintes assuntos: 2.1 - arquitetura de dispositivos móveis, 2.2 - os sistemas operacionais para dispositivos móveis, 2.3 - os aplicativos Android e 2.4 - conceitos de segurança em aplicativos Android.

2.1 Arquitetura de dispositivos Móveis

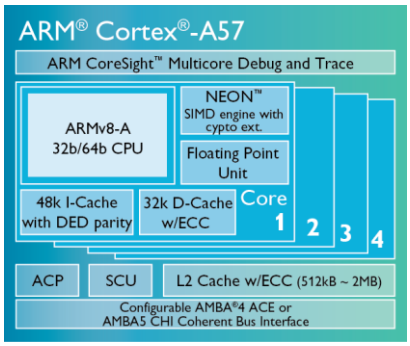
Atualmente boa parte das tarefas feitas em um computador podem ser realizadas em um dispositivo portátil, leve e muito mais cômodo para os usuários. Para que isso fosse possível, toda a arquitetura de um sistema computacional teve que ser repensada, com mais *chips* e ciclos de processamento, além disso, novos sistemas operacionais tiveram que ser construídos, de forma que a eficiência não fosse o fator principal, e sim, que dividisse espaço com consumo energético e interatividade.

De acordo com Auletta (2012) pode-se destacar dois modelos de arquitetura para dispositivos móveis, a Horizontal, usada pela Google e a Vertical, usada pela Apple. A solução vertical é constituída por um sistema operacional próprio, hardware conhecido e um controle muito rígido em seus programas. A horizontal possui uma plataforma aberta, com muitas versões, diversos hardwares produzidos por muitas empresas e um menor controle dos programas fornecidos.

Zanoni (2013) explica que em relação a arquitetura de processadores, a mais difundida para dispositivos móveis é a ARM com os núcleos da série Cortex-A. De acordo com a ARM (2014) há 178 licenças cedidas dessa série para empresas como Samsung, ST-Ericsson, entre outras. A própria empresa apresenta alguns exemplos dessa família de processadores, que podem ser vistos a seguir:

- Processador Cortex-A57: processador de mais alto desempenho da ARM, próxima geração para móvel, empresarial e dispositivos de rede, sua arquitetura está ilustrada na Figura 2;

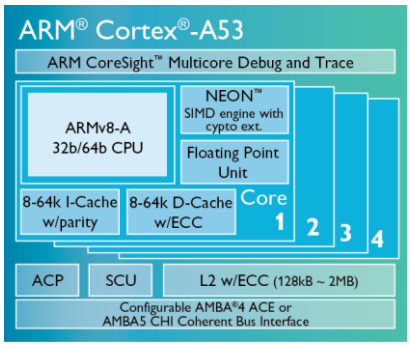
Figura 2: Arquitetura do Cortex- A57.



Fonte: (ARM, 2014).

- Cortex-A53 processador: processador de 64 bits mais eficiente da ARM para aplicações, incluindo móvel, TV digital, Internet das Coisas (IoT) e *networking,* que está ilustrado na Figura 3;

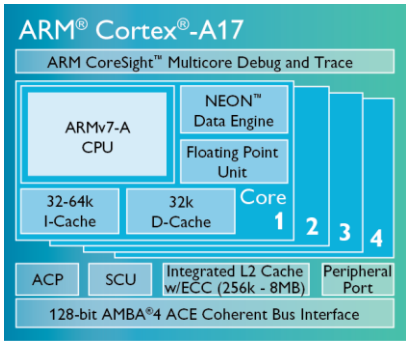
Figura 3: Arquitetura do Cortex-A53.



Fonte: (ARM, 2014).

- Processador Cortex-A17: proporciona ponto de desempenho *mid-range* para qualquer tela, a partir de *smartphones* para Televisores inteligentes, sua arquitetura está ilustrada na Figura 4;

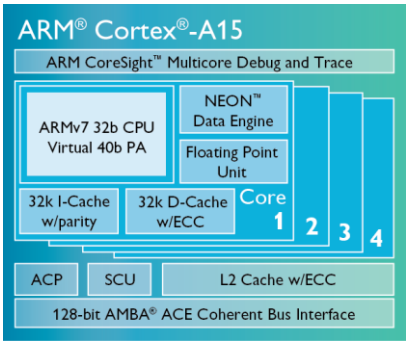
Figura 4: Arquitetura do Cortex-A17.



Fonte: (ARM, 2014).

- Processador Cortex-A15: core de alto desempenho para aplicações exigentes de infraestrutura sem fio atual e móvel, sua arquitetura está ilustrada na Figura 5;

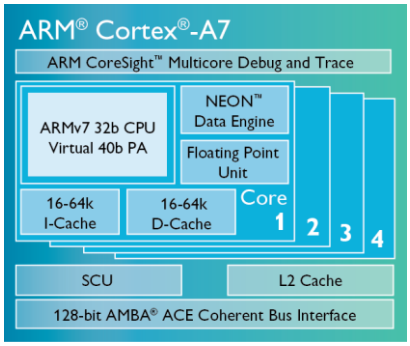
Figura 5: Arquitetura do Cortex-A15



Fonte: (ARM, 2014).

- Processador Cortex-A7: configurações *multi-core* que entregam 800 MHz - 1.2 GHz de frequência típica. O Cortex-A7 que pode ser visualizado na Figura 6 é cada vez mais implantado em dispositivos portáteis;

Figura 6: Arquitetura do Cortex-A7.



Fonte: (ARM, 2014).

As memórias também evoluíram, os primeiros *palmtops* e *smartphones* utilizavam memória *Static Random Access Memory* (SRAM) como memória de armazenamento, essa possuía algumas desvantagens como ser volátil, o que tornava necessário manter a memória energizada, pois caso a bateria estivesse descarregada os dados seriam apagados, além disso, se tratava de uma memória cara, o que aumentava o custo e limitava a quantidade usada nos aparelhos. (HWANG, 2003). A partir disso houve uma rápida migração para a memória Flash[4], que permitia armazenar dados por longos períodos, sem precisar de alimentação elétrica.

[4] Memória de alta performance não volátil.(KINGSTON, 2015).

A migração das memórias *Special Drawing Right* (SDR) para *Double Data Rate* (DDR) ocorreram de forma lenta, a troca só ocorreu definitivamente com a criação da memórias *Synchronous Dynamic Random Access Memory* (SDRAM) destinadas a smartphones, produzidas com técnicas especiais, de forma a utilizarem tensões mais baixas e menos *chips* de *refresh*, o que reduzia o consumo de energia. Inicialmente os *Smartphones* continham memórias pequenas de 64B e 128B, o que era justificável pelo alto custo, mas hoje, é possível encontrar dispositivos populares de 1GB e 2 GB. Os novos projetos de *Smartphones* prometem trazer tamanho de memória similar a de computadores chegando a 8GB, (BABOO, 2015).

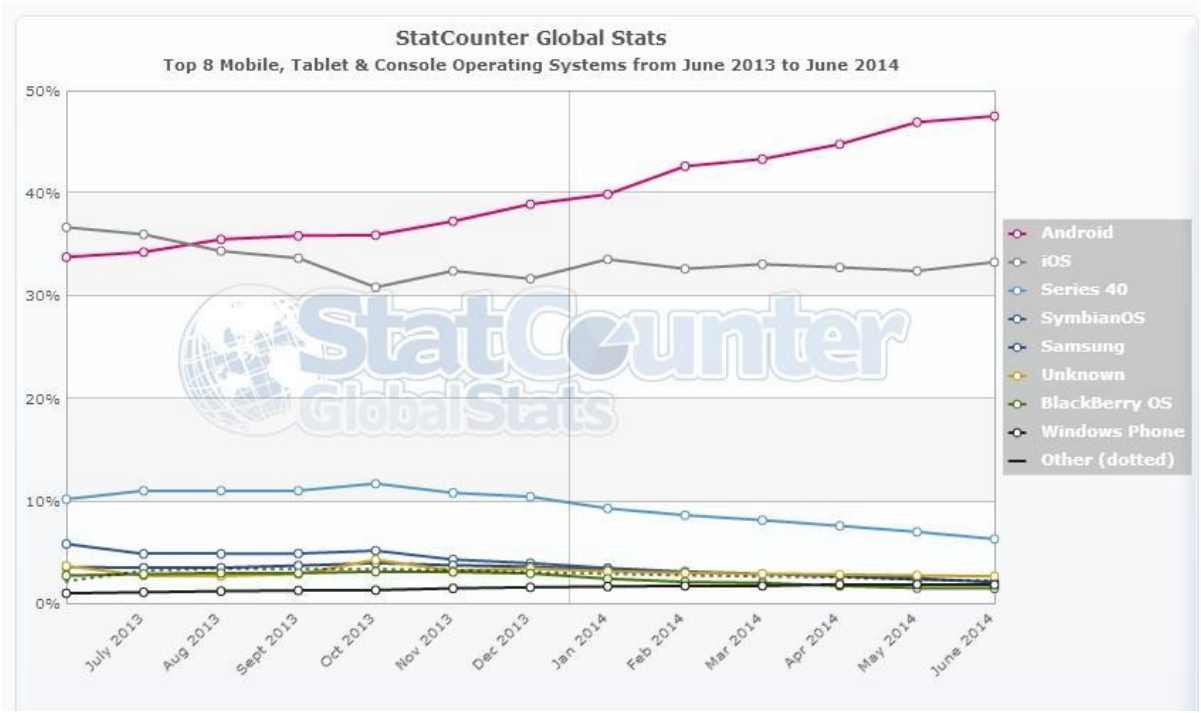
2.2 Sistemas Operacionais para Dispositivos Móveis

Os dispositivos móveis modernos contam com sofisticados sistemas operacionais (SO), como:

- Android: Sistema Operacional da Google, código aberto e amplamente utilizado em diversos aparelhos de diferentes fabricantes, está em sua 5ª versão, o Android 5.0 Lollipop (ANDROID, 2014);
- IOS: Sistema Operacional da Apple, base do iPhone, iPad e iPod *touch*, está em sua 8ª versão, o iOS 8. (APPLE, 2014).

Segundo o Statcounter (2014), o Android é o SO mais usado no mundo, conforme ilustrado na Figura 7.

Figura 7: StatCounter indicando o Android como o SO móvel mais usado no mundo.



Fonte: (STATCOUNTER, 2014).

Em questão de segurança, o Android e IOS têm uma abordagem semelhante em suas lojas, nas quais a segurança dos milhares de aplicativos disponíveis para o usuário é verificada de forma automática e manual. Os dois sistemas isolam os processos em execução em uma caixa de proteção, o que impede que um aplicativo possa tomar o controle de todo o sistema, esse procedimento é feito de forma eficaz através do controle de permissões dos aplicativos, que estabelece a quais dados e recursos do aparelho um aplicativo específico pode acessar. Tanto o Android quanto o IOS têm sistemas de permissões, mas, um pouco diferenciados entre si. O IOS só avisa ao usuário sobre uma solicitação de permissão quando é necessário autorizar o acesso a um determinado recurso, sendo assim, o usuário pode aceitar ou rejeitar cada um deles como quiser, com o App já instalado e funcionando. A Google Play (2014) explica que o Android, relata desde o início e com muitos detalhes as permissões necessárias para um aplicativo, se o usuário não aceitar as condições, o aplicativo não será instalado.

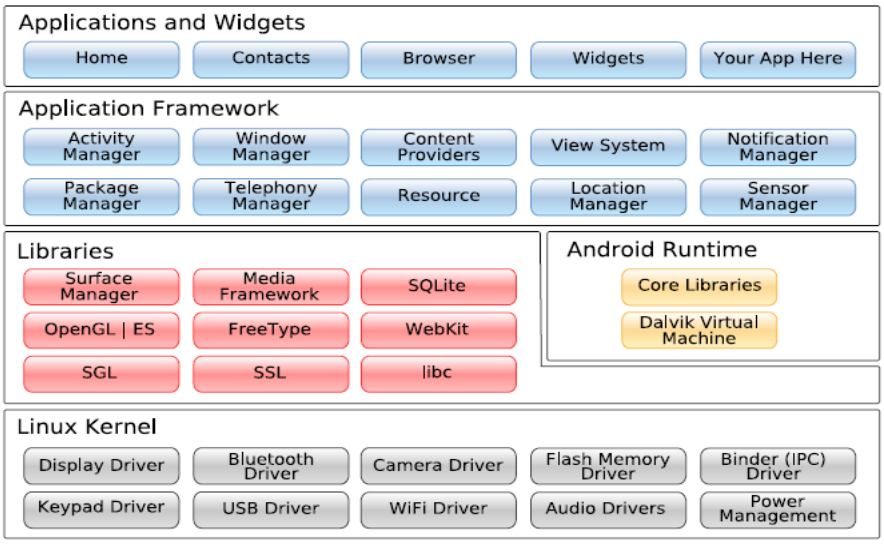
Segundo Schultz *et al.* (2011), o Android tem um grave problema de segurança por parte do controle de permissão na hora da instalação, pois o usuário em sua grande maioria, não presta atenção nas permissões dadas aos aplicativos.

Com base nessas observações e reconhecendo a grande parcela de mercado do Android, esse trabalho destaca e prioriza a realização de um estudo e a apresentação de novas soluções para identificar e apresentar os riscos para o usuário nesse sistema.

2.2.1 O Sistema Operacional Android

O sistema operacional Android é construído e mantido pela Google, tendo seu desenvolvimento em camadas, como mostra a Figura 3, sobre o *Kernel* derivado do Linux.

Figura 3: Camadas da arquitetura Android**.**



Fonte: (BURNETTE, 2009).

Burnette (2009) explica cada camada:

- Linux *Kernel*: fornece alguns serviços essenciais, como segurança, rede e gerenciamento de memória e processos, além de uma camada de abstração de hardware para as outras camadas de *software*.
- *Libraries*: bibliotecas C/C++ utilizadas por vários componentes do sistema, como: bibliotecas para suporte a formatos de áudio, vídeo e imagens; banco de dados *SQLite* entre outros.
- *Application Framework*: fornece funcionalidades necessárias para a construção de aplicativos, através das bibliotecas nativas.

- *Application*: onde se localizam as aplicações.
- Android *Runtime*: onde se encontram as bibliotecas de programação e a máquina virtual Dalvik, onde as aplicações são executadas.

A maior parte dos aplicativos desenvolvidos para Android são escritos em Java usando o *Android Software Development Kit* (SDK), compilados em arquivos *Dalvik*. Os programas também podem ser escritos nas Linguagens C, C++ e Python. Os aplicativos ficam hospedados na Google Play, loja de aplicativos da Google, mas qualquer site pode criar seu próprio repositório. Para submeter aplicativos para Google Play, um desenvolvedor Android precisa primeiramente obter uma conta de desenvolvedor. Após a criação do *Android Package* (APK) o desenvolvedor insere o arquivo na página web da Google Play. Essa página contém meta-informação que mantém o controle de informações referentes ao aplicativo, por exemplo: nome, categoria, versão, tamanho, os preços e suas estatísticas de uso como classificação, número de instalações entre outros.

O sistema Android contém uma defesa contra *malware*, composto de duas partes: o *sandboxing* e o mecanismo de comunicação. No *sandboxing* cada aplicativo alerta o usuário sobre as permissões que estão sendo solicitadas. Cada App é executado com um identificador (ID) de usuário separado, e por padrão não tem permissão para realizar ações ou recursos de acesso que possam ter um efeito adverso sobre o sistema ou em outros aplicativos. Para complementar essa a abordagem *sandboxing* existe o mecanismo de comunicação, quando um usuário baixa um aplicativo através do Google Play, ele é informado sobre as intenções do aplicativo e as permissões solicitadas, além disso, há uma breve explicação sobre o objetivo do uso de cada permissão. Dessa forma fornece-se uma oportunidade final para o usuário avaliar o aplicativo antes de baixar. Para instalá-lo, o usuário deve conceder a todas as permissões solicitadas. Não existem maneiras de baixar um aplicativo negando alguma das permissões.

Contudo, tal defesa usada pelo Android não é eficiente visto que o número de *malwares* está aumentando muito comparado a outros sistemas operacionais móveis como o IOS e o Symbian. Essa informação pode ser vista no relatório 2014 da empresa F-Secure (F-SECURE, 2014), que informa o aparecimento de 277 famílias de *malwares,* sendo que 275 rodam em Android. Esse trabalho esclarece algumas

questões entre *malwares* e aplicativos e apresenta uma proposta para medir o risco de um determinado programa com base nas permissões solicitadas.

2.2.1 Versões Android.

No próprio site oficial do Android (ANDROID, 2014) é possível obter as versões desse sistema operacional e um pouco de sua história.

A primeira versão, o Android 1.0, foi lançada em setembro de 2008 e possuía recursos que na época eram inovadores como Media Player, suporte wi-fi, entre outros. Em fevereiro de 2009 foi lançada uma atualização desse sistema, o Android 1.1 que corrigiu falhas e bugs da versão 1.0. Pouco tempo depois, em abril de 2009 surgiu o Android 1.5, denominado *Cupcake*, sendo o primeiro a receber um apelido de sobremesa, padrão que ainda existe; nessa versão as transições de telas animadas e melhorias no teclado, que passou a funcionar com o dispositivo móvel na horizontal ou vertical foi destaque. Em setembro de 2009 surgiu o Android 1.6, apelidado de *Donut*, nele ocorreu a inclusão da caixa de buscas na tela inicial, o que facilitou pesquisas no próprio dispositivo e na web, além disso, incluiu-se um sistema de síntese de voz. Já em outubro de 2009 foi lançado o Android 2.0, o *Eclair*, que trouxe suporte ao HTML 5 e permitiu a inclusão de várias contas no aparelho, para sincronização de contatos entre fontes diferentes. A versão 2 contou com várias atualizações para correção de bugs, a última foi a 2.3.7 denominada *Gingerbread*, trouxe suporte à resolução *High Definition* (HD) e sensores como barômetro, além de aceitar múltiplas câmeras em um mesmo dispositivo. Em fevereiro de 2011 apareceu o Android 3.0 chamado *Honeycomb*, único sistema operacional desenvolvido exclusivamente para *tablets,* deu suporte a processadores com múltiplos núcleos e permitia a encriptação de todos os dados do usuário. Em outubro de 2011 surgiu a versão 4.0 ou *Ice Cream Sandwich*, uma das novidades dessa versão foi a possibilidade de acessar aplicativos diretamente da tela de bloqueio e desbloqueio por meio de reconhecimento facial, nessa versão também houve atualizações para correção de bugs, como o 4.1 chamado *Jelly Bean* e o *KitKat*, versão 4.4. O último lançamento, até a conclusão desse trabalho, foi no final de 2014, a 5.0 apelidado de *Lollipop* que promete economia de bateria, novos tipos de notificações, melhoria de desempenho atráves do *runtime* ART, possibilidade de

acesso como usuário convidado, que permite a uma terceira pessoa utilizar o dispositivo móvel sem acesso aos dados pessoais do proprietário, entre outras novidades.

2.3 Aplicativos Android

Os aplicativos para Android ou Apps, como são mais conhecidos, utilizam todos os recursos disponíveis no dispositivo, tais como: câmera, GPS, Wifi, microfone, entre outros. Dessa forma é possível fornecer aos usuários Apps para automação de processos e entretenimento, para qualquer idade ou setor. São programados normalmente em Java mas, tal prática não é regra já que o Android suporta muitas outras linguagens como descrito na seção 2.2.1, o uso do Java é justificado pela grande facilidade de manipular as inúmeras *Application Programming Interface* (API) e por trabalhar em uma camada superior em uma máquina virtual.

Ainda em relação aos aplicativos Android, Xu *et al.* (2013) explicam que os mesmos são construídos sob quatro tipos de componentes de aplicativos: as atividades, serviços, provedores de conteúdo e receptores de radiodifusão. Cada um desses tem o seu próprio ciclo de vida. A maior parte dos componentes podem ser iniciados individualmente. Esse trabalho se concentra nos tipos: atividades e componentes de serviço, visto que o objetivo é criar um modelo de análise de risco e um App para aplicação desse modelo. Antes disso, se faz necessário entender componentes básicos de uma aplicação e após, projetá-la. Para tanto, serão apresentadas as subseções: estrutura, virtualização e mercado de aplicativos.

2.3.1 Estrutura dos Aplicativos

Os aplicativos construídos na linguagem Java normalmente contam com o auxílio da interface gráfica em *Xtensible Markup Language* (XML), por ser mais leve e rápida para os dispositivos. Essa trabalha em camadas facilitando o uso do aplicativo com o toque na tela.

Os “aplicativos Android são distribuídos em um formato de arquivo compactado com a extensão“.apk” que contém um arquivo de manifesto “AndroidManifest.xml”, executáveis Dalvik, compilados em classes denominadas

“class.dex”, além de outros recursos arquivos como por exemplo os encontrados dentro da pasta denominada “res”, (XU *et al*, 2013).

A Figura 8 apresenta a estrutura de pastas de um projeto de aplicativo, onde a pasta “Java” contém os arquivos “.class” programados na linguagem Java, o arquivo “res” de acordo com Lecheta (2013) por padrão é dividido em quatro pastas: “*drawable”, “layout”, “menu”, “values”* explicadas a seguir:

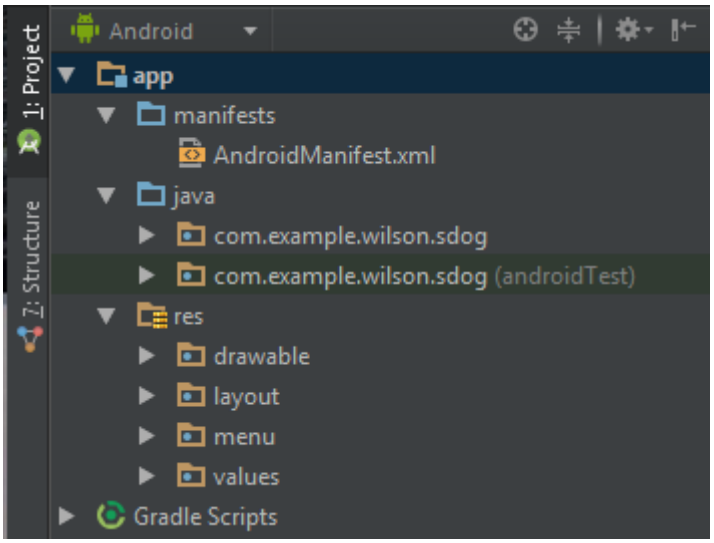
- A pasta *drawable* contém arquivos XML usados para formatação de componentes XML como, por exemplo, botões de caixas de textos. Nessa pasta é possível encontrar ícones e figuras usadas nos componentes de tela XML.

- O repositório layout contém todos os XMLs que ajudam a construir a interface gráfica dos aplicativos, são importadas pelas *class activitys,* usadas para construção de uma nova camada de interfaces interligando o XML com o java.

- A pasta menu contém a formatação dos menus do projeto normalmente presente na parte superior dos aplicativos.

- A pasta *values* é usada para guardar variáveis em XML que podem ser usadas em todo o programa, como variável global. Por padrão, variáveis como o nome do programa e endereço de imagens são salvos nesse arquivo a fim de facilitar a sua alteração.

Figura 8: Projeto de um aplicativo na IDE Android Studio



Fonte: (Próprio autor, 2015).

A pasta *Gradle Scripts* contém arquivos de configuração da IDE de programação e pastas do arquivo res citadas anteriormente. A *Gradle Scripts* dificilmente será alterada pelo programador ficando a cargo apenas da IDE.

2.3.2 Virtualização de aplicativos

Em dispositivos com iOS, os aplicativos são instalados antes de serem executados, essa operação é realizada pelo SO, mas, no Android o processo é diferente. Em razão da grande variedade de aparelhos e diferentes fabricantes que utilizam o sistema operacional da Google, o mesmo precisa trabalhar com um sistema de emulação de máquina virtual. De acordo com MEDNIEKS *et al.* (2012), o Android utiliza uma máquina virtual Dalvik para executar os *softwares* instalados. Essa técnica traz alguns problemas de desempenho, visto que o código tem de ser traduzido para código de máquina toda vez que o programa é executado. Visando solucionar tais problemas, a empresa Google está desenvolvendo a máquina virtual ART.

- Máquina Virtual Dalvik: Os aplicativos instalados no Android são interpretados por essa máquina virtual, após, as informações são enviadas até a interface gráfica. Esse sistema é utilizado desde a versão Android 2.2 até a 4.4 e, permite a execução dos aplicativos em diferentes aparelhos sem a necessidade de edição de código.

- Máquina virtual ART: Introduzido no Android KitKat como uma opção para usuários avançados, o ART pode garantir melhores resultados por trabalhar com o sistema *Ahead-of-Time*. Com essas técnicas os códigos são pré-compilados em linguagem de execução durante o tempo de instalação dos aplicativos. A instalação do App se torna mais lenta, mas, todas as execuções são mais rápidas já que o processo de tradução para linguagem de máquina não seria mais repetido como ocorre em maquinas Dalvik (WANG, 2012).

2.3.3 Mercado de aplicativos

Existem muitos aplicativos no mercado, a maioria são alocados nos repositórios oficiais como a Google Play (GOOGLE PLAY, 2014) e Itunes(ITUNES, 2014) da Apple.

Segundo Braga *et al.* (2012), a Google Play ao contrário da Apple que analisa e verifica regularmente os aplicativos de seu repositório, não revisa todas as aplicações do mercado. No entanto, faz rever algumas aplicações que são filtradas pelo seu modelo de segurança auto definida. Além disso, a Google realiza uma análise automatizada de aplicativos em seu repositório para identificar os potencialmente maliciosos.

Existem outros sites de aplicativos, como por exemplo, a Amazon Appstore para o Android, GetJar, SlideMe Mercado, entre outras, todas apresentam o risco do aplicativo de forma ineficaz. Portões (2014), argumenta que, uma alternativa promissora é apresentar informações sobre o risco relativo ou comparativo. Dessa maneira, os usuários poderiam selecionar aplicativos com base em informações de fácil interpretação e seria uma forma de incentivar os desenvolvedores a solicitarem apenas as permissões necessárias.

2.4 Segurança em Aplicativos Android

Segurança da informação e privacidade são problemas para os usuários de todos os tipos de dispositivos eletrônicos. Chin *et al.* (2012) afirmam que os usuários devem se preocupar mais com a privacidade em seus dispositivos móveis do que em seus computadores, e especialmente se preocuparem com as ameaças de aplicativos maliciosos, já que esses dispositivos são normalmente de uso pessoal.

Como o Android usa o conceito de um ambiente de simulação para separar os recursos de dispositivos dos aplicativos (Ibm, 2014), é possível negar o acesso de um aplicativo a algum recurso, como arquivos e diretórios, a rede de internet, os sensores e as APIs em geral, que não foram devidamente autorizados pelo o usuário no momento da instalação. Essa possibilidade foi herdada do Linux, como segurança em nível de processo, bem como ID de usuários e de grupos com aplicativos e permissões, assim é possível autorizar apenas as operações que o aplicativo tem permissão para realizar. No Linux cada usuário tem seus identificadores e suas autorizações de acesso, no Android essa característica é

voltada para os aplicativos, onde cada um tem o seu ID exclusivo. Por padrão, os aplicativos são executados dentro de um processo básico e em um ambiente de simulação sem permissões, assim, evita-se que os aplicativos acessem o sistema ou recursos. Para que se possa ter acesso aos recursos do sistema, os aplicativos podem solicitar permissões por meio de um arquivo manifesto, que possui função de apresentar o aplicativo.

Os aplicativos podem permitir que seus recursos sejam acessados por outros se tiverem as permissões adequadas. Pode-se compartilhar dados e códigos. Para que isso seja possível é necessário que os aplicativos sejam assinados utilizando a mesma chave privada, após atribui-se o mesmo ID na variável "*android: sharedUserId"*, que está localizado no manifesto do programa.

A assinatura de um aplicativo é o processo de gerar chaves privadas, públicas e certificados de chaves públicas. Com assinatura digital é possível identificar o autor do código e detectar se o aplicativo foi alterado, assim estabelece-se a confiança entre os aplicativos. A possibilidade de compartilhamento de dados e códigos entre aplicativos pode ser vista como um ponto a favor dos programadores, visto que podem ser reaproveitados. Mas, ao usuário, essa possibilidade pode ser um risco, já que um aplicativo sem autorização para conceder determinados recursos pode estar acessando com permissão de outros aplicativos.

Para um melhor entendimento dos principais aspectos de segurança em aplicativos Android, serão apresentadas abaixo as seguintes as subseções: compilação de aplicativos, manifesto, autorizações de operações, vulnerabilidade em dispositivos móveis, malwares para dispositivos móveis, antivírus para Android e análise de risco em aplicativos.

2.4.1 Compilação de Aplicativos

Após concluir a programação do aplicativo é necessário compilá-lo, a fim de gerar o arquivo APK, que pode ser executado em um dispositivo Android. Este processo pode ser executado de duas maneiras: modo *debug* e modo *release,* detalhados a seguir.

- **Aplicativo em *Debug***

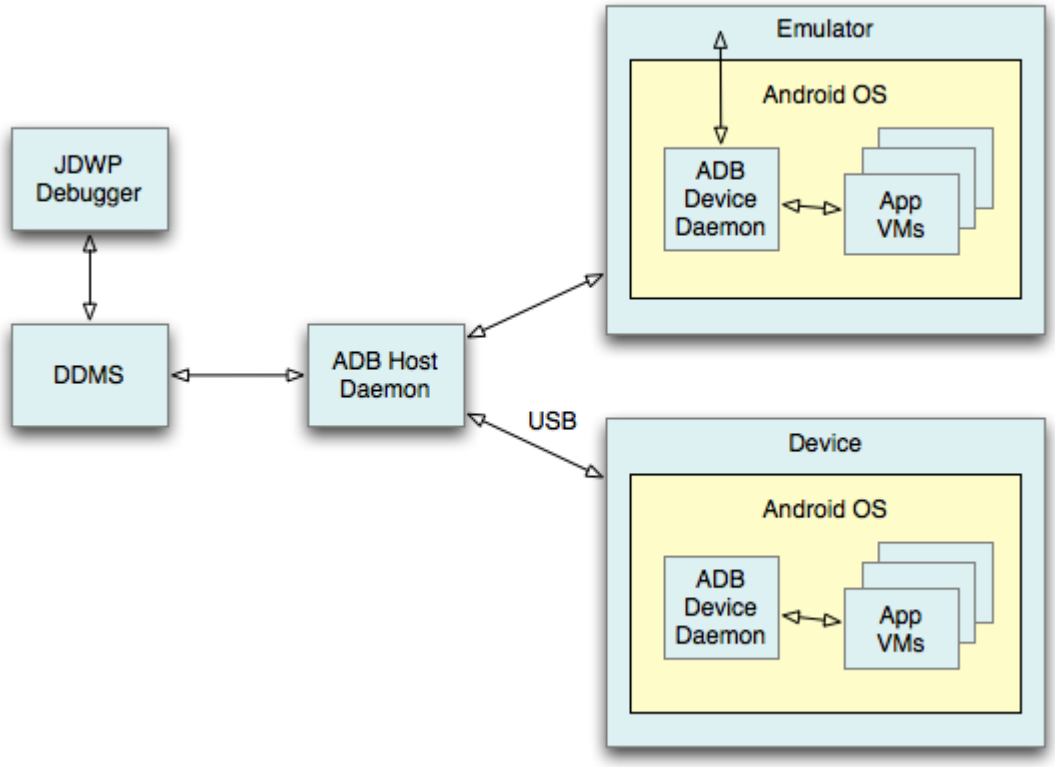
Os aplicativos em *debug* são usados para testes e não devem ser distribuídos; nesse tipo de compilação os programadores tem recursos para testes

do aplicativo, porém não possuem assinatura de proprietário e não são aceitos em repositórios de aplicativos, já que não existe a opção de rastrear o autor do projeto.

Segundo o site Desenvolper (2014), o Android SDK fornece a maioria das ferramentas necessárias para depurar aplicativos. Na depuração é possível visualizar os valores das variáveis e interromper a execução de um aplicativo. No Android Studio um depurador JDWP-*compliant* já está incluído e nenhuma configuração é necessária. Em outras IDE é possível anexar o depurador do Android. Os principais componentes que compõem um ambiente típico de depuração Android são “abd” e *Dalvik Debug Server Monitor* (DDMS) descritos a seguir. A emulação do aplicativo e o funcionamento dos componentes podem ser visualizados na Figura 9.

O “adb” age como um intermediário entre um sistema de desenvolvimento de dispositivos, fornecendo alguns recursos de gerenciamento de emuladores e dispositivos conectados. DDMS é um programa gráfico que se comunica com os dispositivos através do adb, captura *screenshots*, monitora bateria e mensagens.

Figura 9: Mostra como as várias ferramentas de depuração trabalham juntos em um ambiente de depuração típica.



Fonte: (DESENVOLPER, 2014).

Com os componentes devidamente configurados é possível compilar e executar um aplicativo em emulador e analisar possíveis erros, processo importante no desenvolvimento de um *software*.

- **Aplicativo em *Release***

Quando o aplicativo estiver concluído, testado e pronto para ser distribuído comercialmente, ele deve ser assinado com uma identidade única com a finalidade de facilitar a identificação do programa, a versão e os autores, e ainda evitar plágios ou programas maliciosos. Essa assinatura é obrigatória nos repositórios de aplicativos, onde são devidamente investigados e validados.

A assinatura digital é o maior triunfo dos repositórios para garantir a segurança dos usuários, eliminando programas e seus responsáveis dos repositórios após alguma fraude comprovada. Em relação aos *downloads* piratas, essa assinatura pode ser falsificada e a segurança do aplicativo não é garantida.

Para facilitar e baratear os custos de desenvolvimento de um aplicativo, o certificado pode ser auto assinado, ou seja, não há necessidade de uma autoridade de certificação agilizar o processo de publicação do aplicativo.

2.4.2 Manifesto

Como citado anteriormente, é um arquivo em que constam as permissões e informações dos aplicativos, necessárias na hora da instalação do programa. O Android define uma longa lista de permissões de manifesto que protege vários aspectos do sistema ou outros aplicativos. Esse arquivo é fundamental para o projeto, visto que estuda o risco dos aplicativos e pode-se ver a assinatura e permissões do App que indicam suas intenções.

Para solicitar permissões, declara-se um atributo “<user-permission>” no arquivo manifesto como mostra o código abaixo da Figura 10.

Figura 10: Arquivo manifesto.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.wilson.aggggg" >
    <uses-permission android:name="android.permission.READ_SMS" />
    <uses-permission android:name="android.permission.SEND_SMS" />
    <uses-permission android:name="android.permission.SEND_RESPOND_VIA_MESSAGE" />
    <uses-permission android:name="android.permission.write_sms" />
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Fonte: (Próprio autor, 2014).

2.4.3 Autorização de Operações

Quando uma aplicação é instalada no sistema e pretende-se utilizar recursos protegidos do sistema Android, é necessário que os mesmos recursos sejam requisitados adequadamente em permissões. Como já abordado, essas são incluídas no arquivo “manifesto.xml” da aplicação, (CIBRÃO, 2014).

Modelos de Permissões

O modelo de permissões do sistema Android é bastante simples e eficaz, como o sistema possui muitas API para controle do seu hardware, se o aplicativo obtiver a permissão para acessá-las, o recurso é liberado, caso contrário, o recurso é bloqueado. Dentre as API disponíveis para o uso das aplicações estão:

- Câmera: fornece amplo acesso a câmera, pode-se gravar vídeos, tirar fotos, acessar configurações da câmera e editá-las.

- Conectividade: fornece funcionalidades como gestão de ligações, transferência de dados, Wifi e 3G.
- Telefone: fornece acesso a funcionalidades do telefone como realizar ou receber ligações.
- GPS: permite ao aplicativo identificar a sua posição geográfica, além de ligar ou desligar o GPS.

Essas API brevemente descritas são algumas das muitas disponíveis na biblioteca e as quais, o programador tem acesso na construção do seu aplicativo.

Cada API tem mais de uma função, como por exemplo o SMS ou MMS, que permitem o recebimento ou envio de mensagens. Para que fique explicita a intenção do aplicativo para cada função, existe um pedido de permissão, como ilustra a Figura 11, onde as partes destacadas do código são permissões pertinentes a API de SMS ou MMS.

Figura 11: Solicitação de permissões do manifesto.

```
<uses-permission android:name="android.permission.READ_SMS" />
<uses-permission android:name="android.permission.SEND_SMS" />
<uses-permission android:name="android.permission.SEND_RESPOND_VIA_MESSAGE" />
<uses-permission android:name="android.permission.write_sms" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.WRITE_SETTINGS" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.READ_CALENDAR" />
<uses-permission android:name="android.permission.WRITE_CALENDAR" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.GET_ACCOUNTS" />

<application
    android:icon="@drawable/ic_launcher"
    android:label="Security Dog" >
```

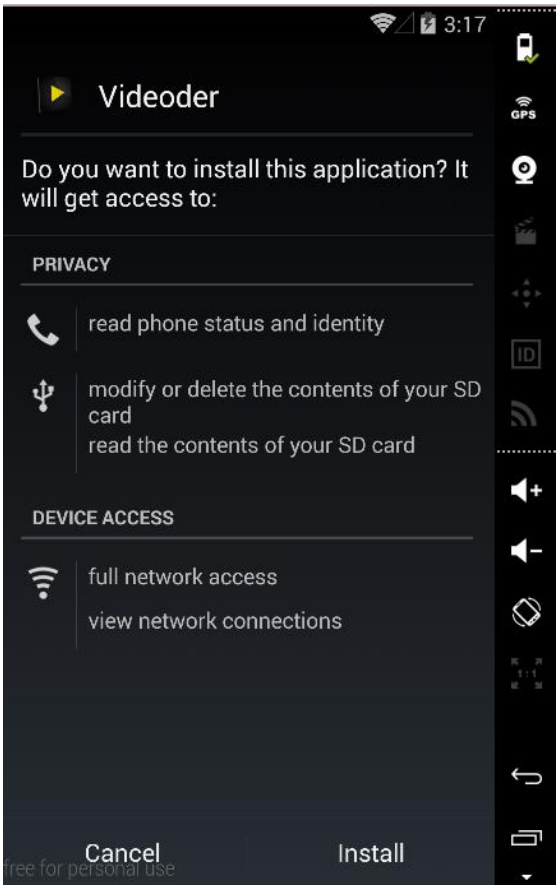
Fonte: (Próprio autor, 2014).

Caso o programador opte por utilizar uma determinada API em seu código, mas não declare ela corretamente, as chamadas das funções podem não surgir efeito e dá-se o erro no processo de execução do App.

2.4.3.1 Concessão de permissões

Quando o usuário faz o *download* de uma aplicação e posteriormente a instalação, aparece uma tela de diálogo onde ele é informado sobre as permissões solicitadas, como ilustra a Figura 12. Quando a instalação é realizada pela loja oficial do Google, a lista de permissões, os detalhes de cada uma e os riscos de concedê-las é apresentada ao usuário antes do *download*.

Figura 12: Instalação de um aplicativo.



Fonte: (Próprio Autor, 2014)

Após dada autorização para o aplicativo não há como voltar atrás e retirar as permissões, nem mesmo editá-las antes da instalação. Existem muitas aplicações como *Adv Permission Manger* e Gerente APK que afirmam gerenciar as permissões, ambas são aplicativos encontrados no repositório da Google, mas, segundo o Projeto Android *Open Source* (2014), esse tipo de aplicação é inadequado, pois funciona interrompendo processos do aplicativo corrompendo dados do dispositivo móvel.

2.4.4 Vulnerabilidades em Dispositivos Móveis

Segundo Cibrão *et al.* (2014), como qualquer sistema, uma plataforma como o Android está sujeito à existência de falhas de seguranças e *bugs*, que introduzem vulnerabilidades. A fácil instalação e o grande acervo de aplicativos, induz o usuário sem preocupação a instalação de muitos aplicativos, sendo vários deles desconhecidos. De acordo com Felt *et al.* (2012), a principal vulnerabilidade do Android é o fácil acesso dos recursos, necessitando apenas da aprovação das permissões na hora da instalação do aplicativo.

Felt *et al.* (2012) explicam ainda que as pessoas não se preocupam com as permissões solicitadas na maioria das vezes, por serem de difícil interpretação. Deve-se ter cuidado na instalação dos apps, pois é nesse momento, de acordo com as permissões requisitadas, que deve-se identificar se o aplicativo é um *malware*.

Um exemplo de vulnerabilidade muito atacada por *malwares* e os próprios usuários a fim de terem o total controle sobre o dispositivo é o rooting, existem várias ferramentas disponíveis no mercado que permitem o rooting de um dispositivo Android, como por exemplo os Exploid e GingerBreak que são usadas para elevação de permissões explorando vulnerabilidades do próprio Linux, e as ferramentas RageAgainstTheCage e ZimperLich que executam o shell do ADB com privilégios de root.

2.4.5 *Malwares* para Dispositivos Móveis

Existem muitos tipos de *malwares* que se aproveitam das mais diversas formas para se propagar e infectar vítimas. A empresa CISCO (CISCO, 2014), afirma que *malwares* podem infectar alvos por serem empacotados com outros aplicativos. Outros podem ser instalados por exploração de uma vulnerabilidade conhecida de uma plataforma móvel, o dispositivo de rede, ou outro *software*. Exemplo disso são os criadores de *malwares* que usam alguma vulnerabilidade de navegador. Delac *et al.* (2011), afirmam que *malware* é projetado tanto para danificar como para interromper um sistema de computador. Essa terminologia é usada para cobrir todo o *softwar*e hostil, incluindo vírus, *Worm*,

Trojan, *Spyware*, *backdoor*, *Rootkit*, e *Botnet*. Peng *et al.* (2014), descrevem os tipos de *malware* como apresentados a seguir:

- Vírus é um tipo de *malware* que entra em um sistema de computador através de hardware ou *software* sem o conhecimento do usuário e, em seguida, atribui-se a um arquivo de programa. O vírus, então, começa a duplicar-se em tarefas mal-intencionadas, no qual foi programado para fazer. A gravidade do vírus inclui os efeitos de dados ou *software* de danos e de negação de serviços.
- *Worm* é um tipo de *malware* que desliza em sistemas de computador sem a permissão do proprietário e opera sem o conhecimento do mesmo. Ao contrário dos vírus, que precisam de intervenção humana para se espalhar, *Worms* podem se espalhar automaticamente a partir de um computador para outro.
- *Spyware* é um tipo de *malware* que coleta informações para fins de publicidade. A presença de *spyware* é normalmente oculto aos usuários, e é difícil de detectar. O *spyware* pode obter números de cartões de crédito, senhas e endereços de e-mail, monitorar atividades web do usuário, verificar os arquivos, criar anúncios pop-up, registrar as teclas digitadas ou alterar a página padrão de navegadores web.
- *Trojan* é um tipo de *malware* com o nome do cavalo de madeira dos gregos, que usavam para se infiltrar em Troy. É uma peça de software nocivo que parece legítimo. Os usuários são normalmente aliciados a executá-lo em seus sistemas. Depois que ele for ativado, ele pode atacar o host irritando os usuários com janelas pop-up ou alterar a sua área de trabalho, excluir arquivos, roubar dados, ou ativar e espalhar outros *malwares*, como vírus.
- *Backdoor* é um programa utilizado para administração remota. Uma vez instalado em um computador, permite que os crackers tenham acesso e controle o hospedeiro através de uma rede local ou da Internet. Um *backdoor* é geralmente capaz de ganhar o controle de um sistema, porque explora processos não documentados no código do sistema.
- *Rootkit* é um tipo especial de *malware* que se esconde em arquivos específicos e processos de ligações de rede em dispositivos

comprometidos. Ele atinge seus objetivos carregando um programa de driver especial ou modificando o *kernel* do sistema operacional.

- *Botnet* é um tipo de *malware* que permite a um atacante controlar remotamente um conjunto de dispositivos comprometidos. Os atacantes muitas vezes os usam para lançar ataques em grande escala de rede, como um ataque distribuído de negação de serviço, spam em massa, ou para coletar informações de privacidade que podem ser usadas para fins ilegais.

2.4.6 Antivírus Android

No sistema operacional Android, os *softwares* antivírus possuem limitações significativas do SO, eles não podem acessar o monitor do sistema de arquivos ou ações em tempo real de aplicativos instalados, como por exemplo, os *downloads* de arquivos maliciosos e outras alterações do sistema de arquivos. Esse fato tem consequências graves para a segurança do dispositivo, pois qualquer App, mesmo sem código malicioso em seu arquivo de pacote com as devidas permissões pode baixar e executar arquivos maliciosos sem serem detectados pelo antivírus. (FEDLER, 2013).

As deficientes soluções de antivírus para a plataforma Android e a falta de métodos de detecção eficazes para o comportamento dinâmico, ou seja, heurística ou abordagens de assinatura baseada em objetos do sistema de arquivos arbitrários, resultam de limitações da plataforma. Segundo o *Projeto Android Open Source* (2014), qualquer *software* instalado em dispositivos Android não pode acessar os diretórios de outros aplicativos, então não é possível analisar arquivos de outros aplicativos e identificar algum comportamento malicioso. Qualquer *malware* não incluído nas bases de dados dos aplicativos antivírus não será detectado. Portanto, considera-se o nível de proteção oferecido pelos antivírus na plataforma Android insuficiente. (FEDLER, 2013).

Segundo Kou *et al.* (2011) algumas técnicas como o uso do Snort[5] que são capazes de analisar o tráfego de rede e identificar ataques, poderiam melhorar a eficiência dos antivírus passando-o de um estado passivo para ativo. Tais práticas

[5] A detecção de intrusão de rede. (SNORT, 2015).

não puderam ser amplamente adotadas devido ao elevado consumo de hardware e energia.

Atualmente algumas soluções de antivírus para os usuários do Android estão sendo fornecidas por empresas, com destaque para: Lookout e Droid Segurança, que são as duas empresas focadas em segurança para dispositivos móveis, F-Secure, Norton e McAfee, que possuem grandes presenças no mercado de *anti-malware*. Embora potencialmente eficaz, o *software anti-malware* é uma solução reacionária, com o tempo médio de 48 dias a partir da liberação de um novo ataque e que o mesmo seja incluído em futuras correções. Uma forma alternativa de detecção de vírus é a solução para desktop baseado em nuvem, além disso estão sendo pesquisadas outras formas para aliviar a sobrecarga dos recursos dos dispositivos portáteis. Uma alternativa de detecção de vírus envolve a revisão de assinaturas de comportamento como o modelo proposto neste trabalho, tal prática não é tão precisa para se afirmar que o aplicativo é ou não um *malware*, a avaliação da aplicação retorna apenas o seu índice de risco e o usuário é que julga se vale apena correr este risco ou não, mas mesmo assim, é a forma mais eficaz, visto que o *malware* pode ser detectado antes da instalação do App.

2.4.7 Análise de Riscos em Aplicativos

Uma maneira de se evitar problemas de segurança é realizando uma avaliação de risco, que consiste em identificar um comportamento malicioso e posteriormente, desinstalá-lo.

As permissões de aplicativos são muito exploradas para análise de risco como explicam Wang *et al.* (2014): em primeiro lugar emprega-se três técnicas de classificação recurso para avaliar o risco de cada permissão. Em segundo lugar, os conjuntos de permissões são avaliados por métodos de seleção de subconjuntos para investigar o risco introduzido pela colaboração de várias permissões. Em terceiro lugar, são analisados os Apps segundo as aplicações de alto risco. Por fim, apresenta-se os riscos aos usuários.

Gates *et al.* (2014) descrevem o modelo PNB que foi construído a partir de um grande conjunto de dados e, em seguida, usado para avaliar um conjunto separado de Apps. Aplicativos foram classificados com base na sua probabilidade de

risco, e em cada percentil mostram o mínimo, máximo e número médio de permissões solicitadas por aplicação. Normalmente, quanto maior o número de permissões, maior o risco. Apesar disso, o risco não é proporcional ao número de permissões. Isso é explicado pelo fato de que as permissões de menor impacto podem ser mais solicitadas, e, várias dessas permissões aumentam o risco relativo. Esse método pode ser eficaz, mas os cálculos probabilísticos em questão estão em constantes atualizações, o que promove um grande volume de processamento, não sendo adequados para análise no dispositivo móvel.

2.4.8 Segurança e Usabilidade de Aplicações de dispositivos Móveis

Chin *et al.* (2012), recomendam a criação de uma nova forma de fornecer indicadores de segurança, e assim, aumentar a confiança do usuário em sua escolha de novos aplicativos e ressaltam que, caso os recursos de segurança sejam complicados os usuários não os utilizarão. Segundo Schultz *et al.* (2011), as interações entre os usuários e os sistemas precisam ser simples. A usabilidade de mecanismos de segurança tem sido estudada em diferentes plataformas móveis. Biddle *et al.* (2009), estabeleceu algumas regras gerais relativas ao conteúdo dos diálogos de segurança como por exemplo: evitar termos desconhecidos, mensagens longas e palavras enganosas ou confusas. Schwarz *et al.* (2011), propuseram que os resultados de pesquisas na web devem ser acompanhados de indicadores para ajudar as pessoas a avaliar o grau de confiabilidade das suas fontes. Eles descobriram que a adição de tais informações a resultados de pesquisa é útil. Todos os estudos sugerem que a apresentação do risco seja em alto nível de forma resumida, especialmente se ele é exibido no início do processo de seleção.

Diederich *et al.* (2013), ressaltam que por questões de segurança e privacidade, a maioria das decisões se relaciona com o risco a que o indivíduo ou o sistema estão ligados, e com a maneira que estes dados estão dispostos para o usuário. Tornou-se habitual conceber julgamentos sobre a percepção do risco e sobre a tomada de decisão de dois modos distintos de pensamento ou sistemas: Sistema 1 e Sistema 2. O Sistema 1 é automático e intuitivo, opera fora da consciência, ao passo que o Sistema 2 requer atenção, é mais lento e lógico que o Sistema 1, porém é mais preciso segundo Kahneman *et al.* (2011). O Sistema 1

mesmo apresentando resultados menos precisos acaba sendo mais aproveitado pelo usuário. De acordo com Finucane *et al.* (2000) isso se deve ao fato de que a tomada de decisões é lógica e automática, enquanto no Sistema 2 os processos contribuem para julgamentos mais deliberados.

Uma das características mais importantes de um sistema é a sua capacidade de facilitar a utilização do mesmo. No entanto, a maioria dos desenvolvedores de *software* apenas se concentram no desenvolvimento sem se preocuparem com sua usabilidade, (REDZUAN *et al.* 2013). Portanto não basta ter um modelo de análise de risco eficiente, a utilização do aplicativo e os resultados da análise de riscos devem ser intuitivos para o usuário. Redzuan *et al.* (2013), descrevem três definições de usabilidade importantes na construção de um sistema: ISO 9241-11, ISO 9126, Jakob Nielsen.

O ISO 9241-11 define usabilidade como a facilidade de uso de sistema ou produto e, fazê-lo de acordo com as necessidades dos usuários, levando em conta os seguintes aspectos:

- Eficácia - Os usuários podem completar uma determinada tarefa e alcançar os objetivos com o produto.
- Eficiência - Medir em tempo o esforço necessário para completar uma tarefa.
- Satisfação - Concentrar no que o usuário sente sobre os produtos e a facilidade de uso.

O ISO 9126, diz que a usabilidade é uma eficiente forma de interação entre interface e usuário. Há quatro fatores mensuráveis no ISO 9126 como indicados a seguir:

- Entendimento - a capacidade do *software* em perceber se suas funções estão de acordo com as necessidades dos utilizadores.
- Apreensibilidade - A capacidade do produto de *software* em facilitar a usabilidade sem a necessidade de estudar a aplicação.
- Operacionalidade - A capacidade do produto de *software* em facilitar a execução e operá-lo.
- Atratividade - A capacidade do produto de *software* em atrair a atenção do usuário e incentivá-lo a ficar com o software.

Outra definição interessante da usabilidade é definida por Jakob Nielsen, ele diz que se trata de uma medida de qualidade da experiência do usuário ao interagir

com um *software*, é um aspecto importante, pois pode influenciar na aceitação do produto. Nielsen especifica cinco atributos de usabilidade:

- Apreensibilidade - O sistema deve não ser complicado para usá-lo.
- Eficiência – O sistema deve ser eficiente ao ser usado, a produtividade deve subir quando o usuário estiver totalmente instruído.
- Memorável - O sistema deve ser simples de memorizar. O usuário não tem que aprender novamente quando volta a usar o sistema.
- Erros – Diminuir o número de erros que o usuário cometa ao usar o programa.
- Satisfação - O sistema deve ser gratificante, tornar os usuários satisfeitos

O modelo a ser construído nesse trabalho levará em conta tais definições de usabilidade, visto que podem influenciar a facilidade de uso e aprendizagem da aplicação, e no grau de satisfação dos usuários.

3 METODOLOGIA

Segundo Barros (2007), metodologia é um estudo de maneiras de abordar determinados problemas com nossos conhecimentos, não buscando soluções, mas fazendo a escolha de formas de achá-las, juntando o conhecimento dos métodos existentes nas diferentes disciplinas científicas ou filosóficas.

No início do trabalho foi realizada uma pesquisa sobre a evolução dos celulares, a arquitetura desses dispositivos e os sistemas operacionais existentes. Optou-se por focar no Android por ser o sistema móvel mais utilizado no mundo, a partir disso estudou-se esse SO a fundo, verificou-se como funciona, as versões, como se dá a segurança dos aplicativos Android, entre outros. Além disso, realizou-se um levantamento de aplicativos na Google Play e descobriu-se o principal problema de segurança relacionado a eles: vulnerabilidades. Esse problema pode ser agravado com grande conectividade dos dispositivos e seus inúmeros sensores, que em controle de alguém mal intencionado pode gerar prejuízo à privacidade. A melhor solução é a prevenção, isto é, instalar apenas aplicativos que não sejam vulneráveis e nem maliciosos. Para isso pode-se analisar o grau de risco de cada programa antes de instalá-lo. Na sequência segue uma proposta com 3 modelos de avaliação de risco, a implementação, os testes e as conclusões.

3.1 Caracterização da pesquisa

Segundo Gil (2002) as pesquisas podem ser classificadas em 3 grupos: descritivas, explicativas e exploratórias, todas são explicadas a seguir.

- Descritivas: tem como objetivo principal a descrição das características de determinada população ou fenômeno ou, o estabelecimento de relações entre variáveis. Uma das características desse tipo de pesquisa é a utilização de técnicas padronizadas de coleta de dados, como por exemplo, o questionário ou a observação sistemática.
- Explicativas: tem como principal objetivo identificar os fatores que determinam ou que contribuem para a ocorrência de fenômenos. Procura explicar a razão e o porquê das coisas.

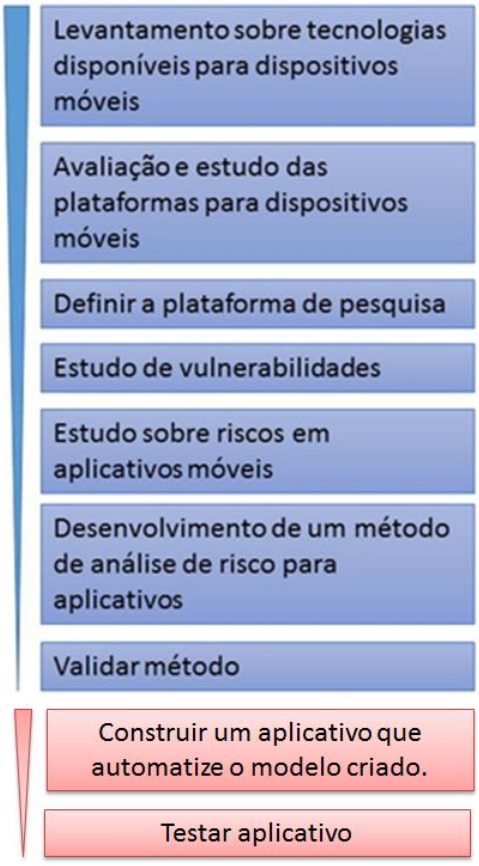
- Exploratórias: tem como objetivo principal proporcionar maior familiaridade com o problema, com vistas a torná-lo mais explícito ou a constituir hipóteses. Visa o aprimoramento de ideias ou a descoberta de intuições.

De acordo com os tipos de pesquisa descritos, esse trabalho se classifica como exploratória, pois tem como objetivo proporcionar maior entendimento sobre o tema em estudo, e propor soluções que resolvam os problemas encontrados.

3.2. Procedimento metodológico

O trabalho está dividido em etapas como ilustrado no fluxograma da Figura 13.

Figura 13: Fluxograma indicando as etapas do trabalho



Fonte: (Próprio Autor, 2014).

A primeira etapa baseia-se, em um estudo teórico sobre o estado da arte em relação ao problema de pesquisa, com o intuito de construir uma base de

conhecimento necessária para a avaliação dos dispositivos móveis e seus sistemas operacionais. Ao final desta etapa almeja-se elencar as tecnologias a serem utilizadas como laboratório base para os experimentos de análise sobre as funcionalidades e níveis de segurança de dispositivos móveis.

Na segunda e terceira etapa será definida a plataforma utilizada para a pesquisa, levando em conta sua importância no mercado de dispositivos móveis. O trabalho ficará restrito em apenas uma plataforma, a fim de apresentar resultados mais específicos e precisos.

A quarta etapa se resume em uma pesquisa sobre vulnerabilidades encontradas nas diferentes funcionalidades do sistema operacional escolhido, e a quinta etapa será pesquisado a identificação dos riscos que o usuário tem ao utilizá-lo.

Na sexta etapa será construída uma métrica para avaliar vulnerabilidades de Apps com a finalidade de encontrar qual apresenta a maior criticidade. Para criação do modelo de análise de risco, primeiramente serão identificadas maneiras de se obter acesso as permissões pedidas por aplicativos já instalados, além disso, serão estudadas as características e riscos de cada permissão, disponibilizadas para os desenvolvedores de Apps. Para isso têm-se como base trabalhos já realizados, como o de Portões *et al.* (2014) e Gates *et al.* (2013), que identificaram quais são as permissões que aparecem com mais frequência em aplicativos maliciosos, chamadas de permissões de risco e a probabilidade de vezes que essas são solicitadas e, posteriormente, atribuir um risco numérico a cada permissão. Nessa etapa serão estudadas maneiras de identificar os riscos para o usuário de forma clara, como Schultz *et al.* (2011) cita ser importante.

A sétima etapa consiste em validar o método de análise de risco desenvolvido. Para atingir esse objetivo serão pesquisadas as aplicações mais baixadas da Google Play e os *malwares* identificados pela empresa SYMANTEC Sistemas, após esses serão analisados pelo modelo, a fim de testá-lo.

O oitavo passo será automatizar o modelo de análise de risco validado, a fim de facilitar a análise de aplicativos instalados nos dispositivos com Android. Esta etapa está entre as mais complexas, pois serão utilizados recursos críticos do Android.

Após a implementação do aplicativo, o último passo será realizar testes a fim de saber se os resultados fornecidos pelo App estão de acordo com os do modelo de risco criado, e se apresenta uma boa usabilidade, requisito para a aceitação do aplicativo pelo público.

4 APRESENTAÇÃO DA PESQUISA E ANÁLISE DOS RESULTADOS

Este capítulo tem como objetivo apresentar a solução proposta após os estudos realizados. Foi possível observar que as maiores falhas de segurança são: o fato de o Android solicitar as permissões necessárias para instalar um aplicativo apenas uma única vez, e a falta de interesse dos usuários em estudar as características e riscos dessas. Sabendo dessas falhas apresenta-se na seção 4.1 o modelo de análise de risco a partir das permissões solicitadas pelos aplicativos, posteriormente na seção 4.2 são apresentados os testes do modelo em diferentes programas, ao final têm-se discussões sobre os resultados obtidos.

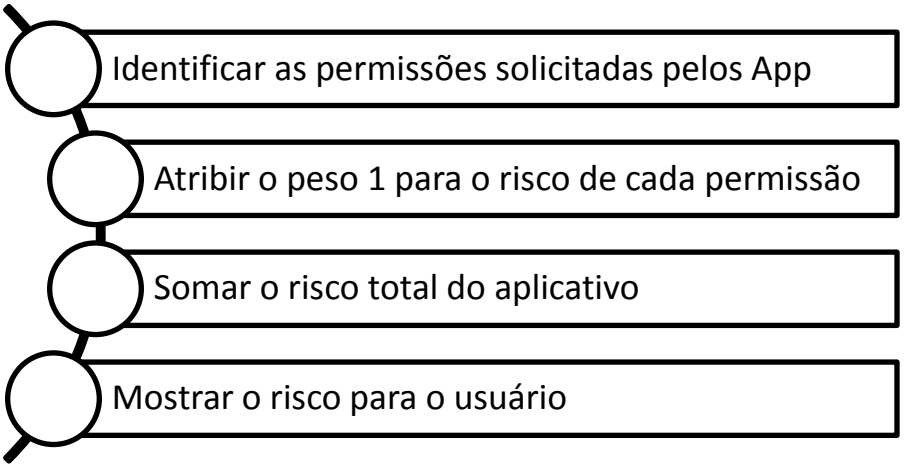
4.1 Modelo Proposto

O modelo criado foi resultado de uma grande pesquisa bibliográfica sobre o assunto, e com a evolução dessa análise bibliográfica foram criados protótipos de modelos capazes de analisar ou não o risco dos aplicativos, para um maior entendimento da evolução do trabalho serão descritos nas subseções a seguir os 3 principais modelos de análise de risco criados, lembrando que este processo é evolucionário, sendo assim os novos modelos criados herdavam processos corretos e corrigiam os erros do modelo anterior.

4.1.1 Modelo 1

Esse é o mais simples e consiste em 4 passos, conforme ilustrado na Figura 9, atribui-se o risco de um aplicativo de acordo com o número de permissões solicitadas.

Figura 14: Fluxograma do modelo de análise de risco 1.



Fonte: (Próprio autor, 2014).

O modelo 1 funciona da seguinte forma:

1. Identifica as permissões solicitadas por cada aplicativo, para isso utiliza-se o programa *Permission Explorer* (GOOGLE PLAY, 2014).
2. Atribui grau de risco 1 para cada aplicativo.
3. Identifica o risco como o somatório do risco de todas as permissões solicitadas.
4. Apresenta o índice de risco para o usuário, de forma que o usuário não necessite de um estudo prévio sobre o aplicativo ou sobre técnicas utilizadas para análise de risco.

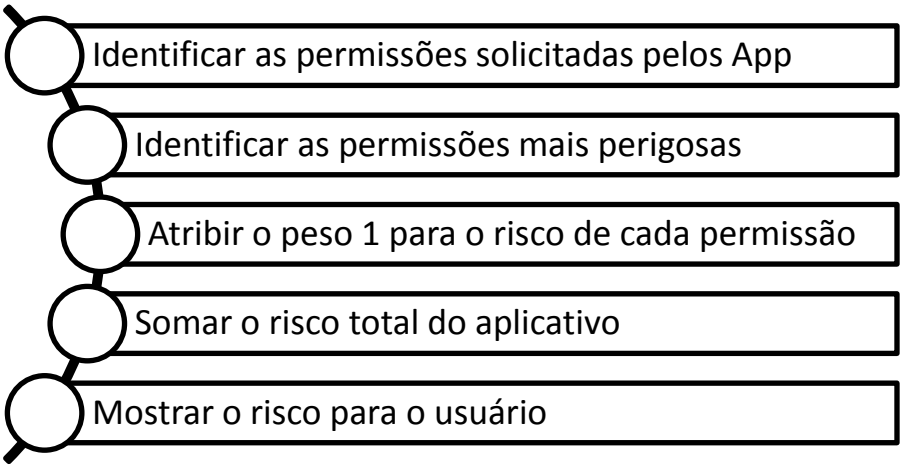
Com os testes preliminares realizados pode-se observar um grave problema, visto que o grau de risco de todas as permissões é o mesmo. Em uma situação hipotética, o aplicativo pede permissão para acessar três recursos locais de leitura do sistema e nenhuma permissão de comunicação, como por exemplo, acesso e leitura de arquivo GPS e a câmera, neste caso, o aplicativo não pode roubar e nem mesmo alterar as permissões, e o risco seria 3. Em outra situação, um aplicativo

pede duas permissões, sendo elas, acesso total a rede e leitura de arquivos, logo, nesse caso o aplicativo poderá ler e transmitir qualquer arquivo armazenado no dispositivo móvel para qualquer lugar do mundo e o risco do aplicativo seria 2. Identificou-se então que na primeira situação relatada, mesmo o aplicativo analisado podendo causar menos danos, o modelo apresenta um risco maior para o usuário, do que na segunda situação. Logo, percebeu-se a necessidade de corrigir esses erros criando um segundo modelo.

4.1.2 Modelo 2

Este modelo foi construído a fim de resolver falhas encontradas nos testes preliminares do modelo 1. Para tanto, houve a necessidade de revisar alguns modelos já conceituados, como por exemplo, o de Gates *et. al.* (2014) que descrevem o modelo PNB, no qual uma das técnicas apresentadas é a separação das permissões por grupos de risco, esse passo será incluído no presente trabalho, mas a classificação dos grupos de risco não será a mesma. O diagrama dos passos para realizar a avaliação de risco com o modelo 2 está ilustrado na Figura 15.

Figura 15: Fluxograma do modelo de análise de risco 2.



Fonte: (Próprio autor, 2014).

Para realizar a análise de risco com o modelo 2 executa-se os seguintes passos:

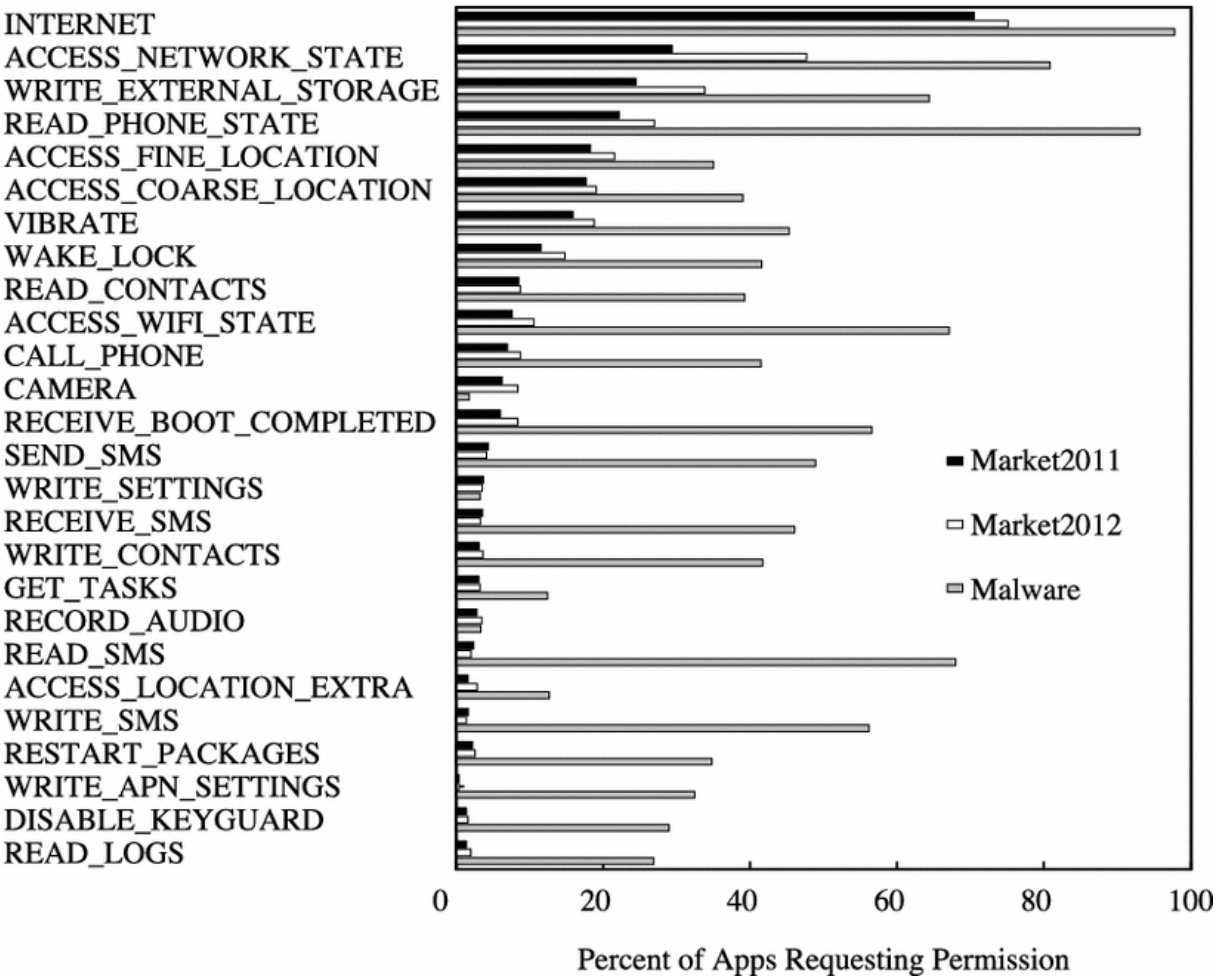
1. Identifica as permissões solicitadas por cada aplicativo, para isso utiliza-se o programa *Permission Explorer*.
2. Identifica entre todas, apenas as permissões de alto risco, presente na maioria dos aplicativos maliciosos de acordo com CHRISTOPHER *et al.* (2014), que identificaram as permissões mais encontradas em aplicativos vulneráveis.
3. Atribui-se o grau de risco 1 para cada aplicativo de alto risco.
4. Apresenta o índice de risco para o usuário, de forma que o usuário não necessite de um estudo prévio sobre o aplicativo ou sobre técnicas utilizadas para análise de risco.

Após testar esse modelo, percebeu-se que a diferença dos graus de riscos entre aplicativos maliciosos e não maliciosos era muito pouca, por exemplo um aplicativo que solicitasse 10 permissões de baixo risco, como nenhum acesso a arquivo e internet resultaria em um risco alto e um aplicativo que solicitasse acesso a internet e a todos os arquivos, o que possibilitaria exportação de arquivos pessoais para terceiros resultaria em risco baixo. A partir disso criou-se o modelo 3.

4.1.3 Modelo 3

A partir da análise dos modelos anteriores, foi construído o modelo 3, com um grau de risco diferente para cada permissão, a fim de ampliar a diferença entre os aplicativos vulneráveis e não vulneráveis. Nesse modelo a atribuição de grau de risco se dará de um modo diferente. Gates et. al. (2014) identificaram o percentual de vezes que uma permissão de alto risco é solicitada por um aplicativo infectado, como observa-se no gráfico da Figura 16, nela existem 3 barras: Preta identificando o percentual de vezes que as permissões são solicitadas pelos aplicativos no ano de 2011, Branca indicando o percentual de vezes que as permissões são solicitadas pelos aplicativos em 2012 e Cinza identificando o percentual de vezes que uma permissão é pedida pelos *malwares* descobertos na plataforma Android.

Figura 16: As permissões mais usadas por malwares.



Fonte: (GATES et. al.,2014).

Utiliza-se esses dados estatísticos para atribuir um grau de risco em cada aplicativo conforme mostrado na Tabela 1. Assim conseguiu-se aumentar o grau de risco dos aplicativos maliciosos.

Tabela 1: Lista das permissões de risco e o seu respectivo risco.

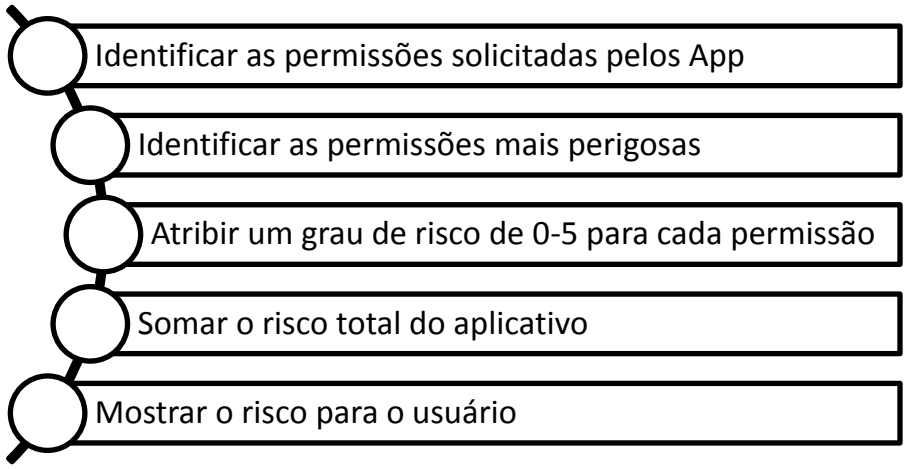
Risco	Apps Maliciosos (%)	Permissões
1	Solicitadas por 01% a 20% dos malwares.	ACESS_LOCATION_EXTRA
		RECORD_AUDIO
		GET_TASKS
		CAMERA
2	Solicitadas por 21% a 40%	READ_LOGS

	dos malwares.	DISABLE_KEYGUARD
		WRITE_APN_SETTINGS
		RESTART_PACKAGES
		READ_CONTACTS
		ACESS_FINE_LOCATION
		ACESS_COARSE_LOCATION
3	Solicitadas por 41% a 60% dos malwares.	WRITE_SMS
		WRITE_SETTINGS
		WRITE_CONTACTS
		RECEIVE_SMS
		SEND_SMS
		VIBRATE
		RECEIVE_BOOT_COMPLETED
		CALL_PHONE
		WAKE_LOCK
4	Solicitadas por 61% a 80% dos malwares.	READ_SMS
		WRITE_EXTERNAL_STORAGE
		ACESS_WIFI_STATE
5	Solicitadas por 81% a 100% dos malwares.	INTERNET
		ACESS_NETWORK_STATE
		READ_PHONE_STATE

Fonte: (Próprio autor,2014)

Com os estudos realizados sobre permissões e seus devidos percentuais de uso em *malware*s, criou-se a versão final do modelo de análise de risco, que está ilustrado no fluxograma da Figura 17.

Figura 17: Fluxograma do modelo de análise de risco 3.



Fonte: (Próprio autor, 2014)

O modelo 3 funciona da seguinte forma:

1. Identifica as permissões pedidas por cada aplicativo, para isso utiliza-se o programa *Permission Explorer*.
2. Identifica entre todas, apenas as permissões de alto risco, presente na maioria dos aplicativos maliciosos de acordo com Christopher *et al.* (2014), que identificaram as permissões mais encontradas em aplicativos vulneráveis.
3. Atribui o grau de risco de 0-5 para cada aplicativo de alto risco de acordo com a Tabela 1.
4. Soma-se os riscos de todas as permissões encontradas.
5. Apresenta-se o índice de risco para o usuário, de forma que o usuário não necessite de um estudo prévio sobre o aplicativo ou sobre técnicas utilizadas para análise de risco.

Mostrar o risco apenas de forma numérica poderia ser confuso para o usuário, pensando nisso, os riscos serão representados em uma barra de progresso. O valor máximo é a soma do grau de risco de todas as permissões alto risco.

Além disso, serão acrescentados 3 estados de risco: baixo, médio e alto, identificados pela cor da barra de progresso.

- Baixo: entre 0% até média de risco dos aplicativos não infectados analisados. Representado pela cor verde.

- Médio: a média de risco dos aplicativos não infectados até a média de risco dos *malware*s analisados. Representado pela cor amarela.
- Alto: acima da média do risco dos *malware*s analisados até 100%. Representado pela cor vermelha.

4.2 Testes dos Modelos de Análise de Risco

Para avaliar, de forma preliminar os modelos propostos, foram realizados testes. Esses são constituídos de 4 etapas, conforme descrito a seguir:

- Foram levantadas 5 ferramentas da Google Play, dentre elas as 4 mais baixadas e 1 que já foi atacada.
- Foram elencados 5 *malware*s descobertos na plataforma Android, a fim de fazer uma comparação.
- Os 5 aplicativos e os 5 *malwares* foram analisados com os 3 modelos de análise de risco, como ilustram as figuras abaixo.
- Os modelos foram comparados levando em consideração a taxa de acerto, pois não é possível comparar o valor do risco atribuído entre os modelos já que os mesmos são dados de forma diferente.

A primeira análise de risco é do *malware* DroidDream, o processo foi repetido para os três modelos e os resultados encontram-se ilustrados na Figura 18. Pode-se observar que o modelo 1 identificou o grau de risco como sendo 12, o modelo 2 identificou o grau de risco como sendo 11 e o modelo 3 identificou o grau de risco como sendo 39, logo o último modelo apresentou o maior risco dentre os demais.

Figura 18: Análise do malware DroidDream.

Malware	Permissões	Modelo 1	Modelo 2	Modelo 3
Android.DroidDream Pode ser empacotado em outro aplicativo para realizar comandos no dispositivo infectado.	android.permission.INTERNET	1	1	5
	android.permission.READ_PHONE_STATE	1	1	5
	android.permission.WRITE_EXTERNAL_STORAGE	1	1	4
	android.permission.ACCESS_NETWORK_STATE	1	1	5
	android.permission.WAKE_LOCK	1	1	3
	android.permission.ACCESS_COARSE_LOCATION	1	1	3
	android.permission.ACCESS_FINE_LOCATION	1	1	2
	android.permission.READ_CONTACTS	1	1	2
	android.permission.WRITE_CONTACTS	1	1	3
	android.permission.WRITE_SETTINGS	1	1	3
	android.permission.CHANGE_WIFI_STATE	1	0	0
	android.permission.ACCESS_WIFI_STATE	1	1	4
Risco Total		12	11	39

Fonte: (Próprio Autor, 2014).

Prosseguindo com as testagens, a Figura 19 descreve a análise de risco do *malware* Walkinwat/Pirater. Na realização dessa análise o modelo 3 também apresentou o maior risco: 34, enquanto o modelo 2 apresentou o risco menor: 11 e modelo 1 identificou o grau de risco como sendo 13.

Figura 19: Malware Pirater relatados pela SYMANTE Sistemas.

Malware	Permissões	Modelo 1	Modelo 2	Modelo 3
Android.Walkinwat/Pirater Envia SMS para toda a lista de contatos e rouba informações pessoais.	android.permission.CAMERA	1	1	1
	com.android.vending.CHECK_LICENSE	1	0	0
	android.permission.ACCESS_FINE_LOCATION	1	1	2
	android.permission.ACCESS_COARSE_LOCATION	1	1	3
	android.permission.INTERNET	1	1	5
	android.permission.ACCESS_NETWORK_STATE	1	1	5
	android.permission.VIBRATE	1	1	3
	android.permission.SEND_SMS	1	1	3
	android.permission.READ_CONTACTS	1	1	2
	android.permission.READ_PHONE_STATE	1	1	5
	android.permission.MODIFY_PHONE_STATE	1	0	0
	android.permission.CALL_PHONE	1	1	3
	android.permission.READ_LOGS	1	1	2
Risco Total		13	11	34

Fonte: (Próprio Autor, 2014).

Logo após, realizou-se a análise de risco do *malware* RootSmart, que está detalhada na Figura 20. De acordo com os resultados obtidos, pode-se observar que o modelo 3 apresentou um risco 41, o modelo 2 apresentou um risco 16 e o modelo 1 apresentou risco 37. Ressalta-se que os riscos identificados não devem ser comparados entre eles.

Figura 20: Análise do Malware RootSmart relatados pela SYMANTE Sistemas.

Malware	Permissões	Modelo 1	Modelo 2	Modelo 3
Android.RootSmart / Bmaster Este malware se aproveita da GingerBreak para explorar e obter privilégios de root.	android.permission.ACCESS_WIFI_STATE	1	1	4
	android.permission.CHANGE_WIFI_STATE	1	0	0
	android.permission.BLUETOOTH	1	0	0
	android.permission.BLUETOOTH_ADMIN	1	0	0
	android.permission.WRITE_APN_SETTINGS	1	1	2
	android.permission.READ_SYNC_SETTINGS	1	0	0
	android.permission.WRITE_SYNC_SETTINGS	1	1	1
	android.permission.GET_ACCOUNTS	1	0	0
	android.permission.VIBRATE	1	1	3
	android.permission.FLASHLIGHT	1	0	0
	android.permission.HARDWARE_TEST	1	0	0
	android.permission.WRITE_SECURE_SETTINGS	1	1	1
	android.permission.READ_SECURE_SETTINGS	1	0	0
	android.permission.CAMERA	1	1	1
	android.permission.MODIFY_PHONE_STATE	1	0	0
	android.permission.READ_PHONE_STATE	1	1	5
	android.permission.INTERNET	1	1	5
	android.permission.RECEIVE_BOOT_COMPLETED	1	0	0
	android.permission.SYSTEM_ALERT_WINDOW	1	0	0
	android.permission.GET_TASKS	1	1	1
	android.permission.CHANGE_CONFIGURATION	1	0	0
	android.permission.WAKE_LOCK	1	1	3
	android.permission.DEVICE_POWER	1	0	0
	android.permission.ACCESS_FINE_LOCATION	1	1	2
	com.android.launcher.permission.INSTALL_SHORTCUT	1	0	0
	com.android.launcher.permission.UNINSTALL_SHORTCUT	1	0	0
	android.permission.WRITE_EXTERNAL_STORAGE	1	1	4
	android.permission.ACCESS_NETWORK_STATE	1	1	4
	android.permission.RESTART_PACKAGES	1	1	2
	android.permission.DELETE_CACHE_FILES	1	0	0
	android.permission.ACCESS_CACHE_FILESYSTEM	1	0	0
	android.permission.READ_OWNER_DATA	1	0	0
	android.permission.WRITE_OWNER_DATA	1	0	0
	android.permission.WRITE_SECURE_SETTINGS	1	0	0
	android.permission.WRITE_SETTINGS	1	1	1
	android.permission.MOUNT_UNMOUNT_FILESYSTEMS	1	0	0
	android.permission.READ_LOGS	1	1	2
Risco Total:		37	16	41

Fonte: (Próprio Autor, 2014).

Prosseguindo, o *Malware* Zeahache também foi analisado com os modelos 1,2 e 3, os riscos identificados são 21, 11 e 40 respectivamente, como é possível constatar na Figura 21.

Figura 21: Análise do Malware Zeahache relatados pela SYMANTE Sistemas.

Malware	Permissões	Modelo 1	Modelo 2	Modelo 3
Android.Zeahache Cavalo de Tróia que eleva privilégios de aplicações, descoberto em um aplicativo de língua chinesa disponível para download na App da google.	android.permission.ACCESS_NETWORK_STATE	1	1	4
	android.permission.ADD_SYSTEM_SERVICE	1	0	0
	android.permission.READ_CONTACTS	1	1	3
	android.permission.READ_SMS	1	1	4
	android.permission.SEND_SMS	1	1	3
	android.permission.WRITE_SMS	1	1	3
	android.permission.RECEIVE_SMS	1	1	3
	android.permission.INTERNET	1	1	5
	android.permission.WRITE_APN_SETTINGS	1	1	2
	android.permission.ADD_SYSTEM_SERVICE	1	0	0
	android.permission.CHANGE_NETWORK_STATE	1	0	0
	android.permission.CALL_PHONE	1	1	3
	android.permission.MODIFY_PHONE_STATE	1	0	0
	android.permission.READ_PHONE_STATE	1	1	5
	com.android.launcher.permission.INSTALL_SHORTCUT	1	0	0
	com.android.launcher.permission.UNINSTALL_SHORTCUT	1	0	0
	android.permission.MODIFY_AUDIO_SETTINGS	1	0	0
	android.permission.PROCESS_OUTGOING_CALLS	1	0	0
	android.permission.CHANGE_WIFI_STATE	1	0	0
	android.permission.ACCESS_WIFI_STATE	1	0	4
	android.permission.GET_TASKS	1	1	1
Risco Total:		21	11	40

Fonte: (Próprio Autor, 2014).

O *Malware* BgServ, assim como os demais foi analisado com os 3 modelos e mesmo solicitando poucas permissões, o modelo 3 indicou risco 43, o modelo 1 apontou risco 13 e o modelo 2 risco 12. Esses resultados podem ser observados na Figura 22.

Figura 22: Análise do Malware BgServ relatados pela SYMANTE Sistemas.

Malware	Permissões	Modelo 1	Modelo 2	Risco 3
Android.BgServ Trojanized versão da ferramenta Android Market divulgado pelo Google, em março, o sexto, para eliminar os efeitos do DroidDream. O trojan transmite as informações do dispositivo para um local remoto. Ela mostra mais do que nunca falhas de segurança da google Market, tendo 5.000 usuários afetados.	android.permission.ACCESS_FINE_LOCATION	1	1	2
	android.permission.ACCESS_COARSE_LOCATION	1	1	2
	android.permission.ACCESS_NETWORK_STATE	1	1	4
	android.permission.WRITE_EXTERNAL_STORAGE	1	1	4
	android.permission.RECEIVE_BOOT_COMPLETED	1	1	4
	android.permission.RECEIVE_SMS	1	1	3
	android.permission.SEND_SMS	1	1	3
	android.permission.ACCESS_NETWORK_STATE	1	1	4
	android.permission.CHANGE_NETWORK_STATE	1	0	0
	android.permission.READ_PHONE_STATE	1	1	5
	android.permission.WAKE_LOCK	1	1	3
	android.permission.WRITE_EXTERNAL_STORAGE	1	1	4
	android.permission.INTERNET	1	1	5
Risco Total		13	12	43

Fonte: (Próprio Autor, 2014).

Após os cinco malwares serem analisados, fez-se necessário averiguar cinco aplicativos seguros, para que fosse possível construir uma análise de cada método entre os malwares e apps. Os aplicativos averiguados foram: Pou, Facebook, Jogo Snake, Hungry Snake e Gmail, onde todos os resultados obtidos serão relatados a seguir.

A análise dos apps começa com o aplicativo Pou, de acordo com a Figura 23. Pode-se observar que o número de permissões solicitadas de risco é menor do que as dos *malwares* analisados acima, pois o modelo 2 apontou para o risco 5, o que é abaixo de todos os malwares analisados, o risco do modelo 3 foi 18 e do modelo 1 foi 9.

Figura 23: Análise de risco do aplicativo Jogo Pou, um dos jogos mais baixados da Google Play.

Aplicativo	Permissões	Modelo 1	Modelo 2	Modelo 3
Jogo POU Você sabe o que é preciso para cuidar de seu próprio animal de estimação alienígena?! Alimentá-lo, limpá-lo, brincar com ele e vê-lo crescer ao mesmo tempo em que você conquista novos níveis e desbloqueia papéis de parede e roupas transadas à sua escolha.	android.permission.ACCESS_WIFI_STATE	1	0	0
	android.permission.ACCESS_NETWORK_STAT	1	1	5
	android.permission.BLUETOOTH	1	0	0
	android.permission.BLUETOOTH_ADMIN	1	0	0
	android.permission.WAKE_LOCK	1	1	3
	android.permission.INTERNET	1	1	5
	android.permission.RECORD_AUDIO	1	1	1
	android.permission.WRITE_EXTERNAL_STORA	1	1	4
	android.permission.READ_EXTERNAL_STORA(	1	0	0
Risco Total		9	5	18

Fonte: (Próprio Autor, 2014).

O aplicativo do Facebook é um dos mais baixados da Google Play, e como se trata de uma rede social interativa, necessita de muitas permissões, conforme ilustrado na Figura 24. A análise desse aplicativo com os modelos 1, 2 e 3 forneceu os resultados 19, 11 e 30 respectivamente. Apesar dos modelos terem oferecido um risco elevado para um App popular como o Facebook, isso não significa que eles não funcionem de forma eficaz, pelo contrário, aplicativos como os de redes sociais normalmente solicitam acesso a recursos críticos, o que pode prejudicar a segurança do usuário caso fique em mãos de pessoas mal intencionadas. Portanto, a indicação de um risco elevado não quer dizer que um aplicativo não possa ser instalado, e sim indica que o usuário deve ter mais cautela, verificar fontes e ser mais atencioso em relação a atualizações.

Figura 24: Análise do aplicativo Facebook, uma das redes sociais mais baixadas da google play.

Aplicativo	Permissões	Modelo 1	Modelo 2	Modelo 3
Facebook Manter-se em contato com os amigos está mais rápido que nunca, • Veja o que os amigos estão fazendo • Compartilhe atualizações, fotos e vídeos • Receba notificações quando amigos curtem e comentam suas publicações • Envie mensagens de texto, participe de bate-papos e conversas em grupo • Jogue jogos e use seus aplicativos favoritos Agora você pode obter acesso antecipado à próxima versão do Facebook para Android se tornando um beta tester.	android.permission.ACCESS_COARSE_LOCAT	1	1	2
	android.permission.ACCESS_WIFI_STATE	1	1	4
	android.permission.WRITE_SYNC_SETTINGS	1	0	0
	android.permission.CONTACTS	1	0	0
	android.permission.WRITE_EXTERNAL_STORA	1	1	4
	android.permission.AUTENTICATE_ACCOUNTS	1	0	0
	android.permission.READ_SYNC_SETTINGS	1	0	0
	android.permission.WAKE_LOCK	1	1	3
	android.permission.GET_ACCOUNTS	1	0	0
	android.permission.VIBRATE	1	1	3
	android.permission.MANAGE_ACCOUNTS	1	0	0
	android.permission.READ_CONTACTS	1	1	2
	android.permission.WRITE_CONTACTS	1	1	3
	android.permission.ACCESS_FINE_LOCATION	1	1	2
	android.permission.CAMERA	1	1	1
	android.permission.READ_EXTERNAL_STORA	1	0	0
	android.permission.READ_PHONE_STATE	1	0	0
	android.permission.INTERNET	1	1	5
	android.permission.RECORD_AUDIO	1	1	1
Risco Total		19	11	30

Fonte: (Próprio Autor, 2014).

A análise do Jogo Snake está ilustrada na Figura 25, sabendo-se que a versão do jogo escolhida para o presente trabalho já foi atacada por hackers, a análise foi efetuada na versão original, livre de ameaças. O modelo 1 e 2 identificaram risco 6, já o modelo 3 identificou risco 26, logo, podemos concluir que o aplicativo pede muitas permissões de alto risco, o que pode ser um atrativo para hackers, já que a alteração do código não modifica as permissões solicitadas, o que dificulta a detecção do aplicativo como sendo malicioso.

Figura 25: Analise de um aplicativo oficial que já foi infectado.

Aplicativo	Permissões	Modelo 1	Modelo 2	Modelo 3
Snake Snake é um remake do clássico jogo da serpente para o Android, porém já foi alvo de malwares.	android.permission.ACCESS_NETWORK_STATE	1	1	5
	android.permission.ACCESS_WIFI_STATE	1	1	4
	android.permission.INTERNET	1	1	5
	android.permission.READ_EXTERNAL_STORAGE	1	1	4
	android.permission.READ_PHONE_STATE	0	0	0
	android.permission.WAKE_LOCK	1	1	3
	android.permission.WRITE_EXTERNAL_STORAGE	1	1	5
Risco Total		6	6	26

Fonte*: (Próprio Autor, 2014).*

De acordo com a análise de risco do aplicativo Hungry Snake, apresentada na Figura 26, os modelos 1,2 e 3 apontam os riscos 3,3 e 13 respectivamente.

Figura 26: Análise de risco do aplicativo Hungry Snake, o jogo nunca infectado mais baixado da categoria.

Aplicativo	Permissões	Modelo 1	Modelo 2	Modelo 3
Hungry Snake Serpente Jogo de cobra clássico mais baixado, pode comer maçãs, pegar itens especiais, criar cobra mais e bater as pontuações! Sete mapas e três peles de cobra para desbloquear! Completamente grátis! Mais níveis e as características em breve! Tudo está bem com os controles, por favor leia atentamente as instruções ou mudar o tipo de controle nas configurações.	android.permission.ACCESS_NETWORK_STATE	1	1	5
	android.permission.INTERNET	1	1	5
	android.permission.VIBRATE	1	1	3
Risco Total		3	3	13

Fonte: (Próprio Autor, 2014).

Enfim, a análise do aplicativo Gmail pode ser observada na figura 27, e retornou os seguintes resultados: risco 11 com o modelo 1, risco 6 com o modelo 2 e risco 22 com o modelo 3.

Figura 27: Análise de risco do aplicativo do GMAIL, está entre as mais baixadas e recomendadas da Google Play.

Aplicativo	Permissões	Modelo 1	Modelo 2	Modelo 3
GMAIL O Gmail foi concebido com base na ideia de que o e-mail pode ser mais intuitivo, eficiente e útil. E até mesmo mais divertido. Receba seus e-mails instantaneamente através de notificações de envio, leia e responda às suas conversas on-line e off-line e pesquise e localize qualquer e-mail. O Gmail também permite: • Gerenciar várias contas • Visualizar e salvar anexos • Configurar notificações de marcadores	android.permission.ACCESS_NETWORK_STATE	1	1	5
	android.permission.INTERNET	1	1	5
	android.permission.MANEGE_ACCAUNTS	1	0	0
	android.permission.NFC	1	0	0
	android.permission.READ_CONTACTS	1	1	2
	android.permission.READ_EXTERNAL_STORAGE	1	0	0
	android.permission.READ_SYNIC_STATE	1	0	0
	android.permission.USE_CREDENTIALS	1	0	0
	android.permission.VIBRATE	1	1	3
	android.permission.WAKE_LOCK	1	1	3
	android.permission.READ_EXTERNAL_STORAGE	1	1	4
Risco Total		11	6	22

Fonte: (Próprio Autor, 2014).

Em consonância com as análises realizadas e apresentadas acima, podemos visualizar a evolução dos métodos criados, isso só foi possível pelo esforço de tentar identificar o comportamento das permissões, e ainda a relação das mesmas com os riscos do aplicativo. Como já citado anteriormente, o grau de risco não é suficiente para comparar os modelos, já que para os três são atribuídos pesos e meios diferentes para cada aplicativo. A fim de melhorar a avaliação entre os modelos e facilitar o entendimento ao usuário, apresenta-se a seguir mais um passo na análise dos resultados:

O risco será apresentado como um conceito, os quais são: baixo, médio e alto, e sua classificação dependerá das seguintes regras:

- Para ter um risco Baixo o App deve ter um risco abaixo da média de risco dos aplicativos não maliciosos;
- O risco Médio será atribuído aos aplicativos cujo valor numérico for maior que a média de risco dos aplicativos não maliciosos e menor ou igual a média de risco dos *malwares*.
- O aplicativo terá um risco Alto, se o risco do App for maior que a média do risco dos *malwares*.

As faixas para divisão dos riscos podem ser observadas na Tabela 2.

Tabela 2: Médias dos riscos analisados.

Modelos Testados	Média do Risco dos Malwares Analisados	Média do Risco dos Apps Analisados
Modelo 3	Risco 39	Risco 22
Modelo 2	Risco 12	Risco 6
Modelo 1	Risco 13	Risco 9

Fonte:(Próprio Autor, 2014).

Após obter as faixas para divisão de riscos para os três modelos, é possível identificar o conceito do risco dos aplicativos e malwares, como mostrado nas Figuras 28 e 25, que apresentam os resultados dos testes em aplicativos maliciosos e não maliciosos respectivamente.

Figura 28: Resultados dos testes para os aplicativos maliciosos.

Fonte: (Próprio Autor, 2014).

De acordo com a figura acima, o modelo 3 retornou 2 aplicativos de risco médio, 3 de risco alto e nenhum de risco baixo, apresentando portando os melhores resultados para esse teste.

A Figura 29 apresenta os resultados da análise de risco para os aplicativos não maliciosos. Nesses testes o modelo 3 também apresentou bons resultados, pois obteve-se 3 aplicativos com o risco baixo, 2 aplicativos com o risco médio e nenhum com risco alto. Os aplicativos identificados com risco médio foram o jogo Snake e o App do Facebook.

Figura 29: Resultados dos testes para aplicativos não maliciosos.

Fonte: (Próprio autor, 2014).

Os usuários devem ficar atentos com os aplicativos de risco médio, pois nesse caso não é possível afirmar que se trata de um malware, mas caso caia em mãos de pessoas mal intencionadas, o usuário pode vir a ter problemas de segurança. Como o Facebook, um dos aplicativos mais baixados da Google Play, foi identificado com risco médio, podem vir a surgir desconfianças sobre a originalidade dos resultados, ou ao correto funcionamento dos modelos, mas, como já citado, os modelos desenvolvidos não visam afirmar se o aplicativo é ou não um malware, o objetivo é informar ao usuário sobre os riscos. É o usuário quem decide se vale a pena ou não instalá-lo em seu dispositivo móvel.

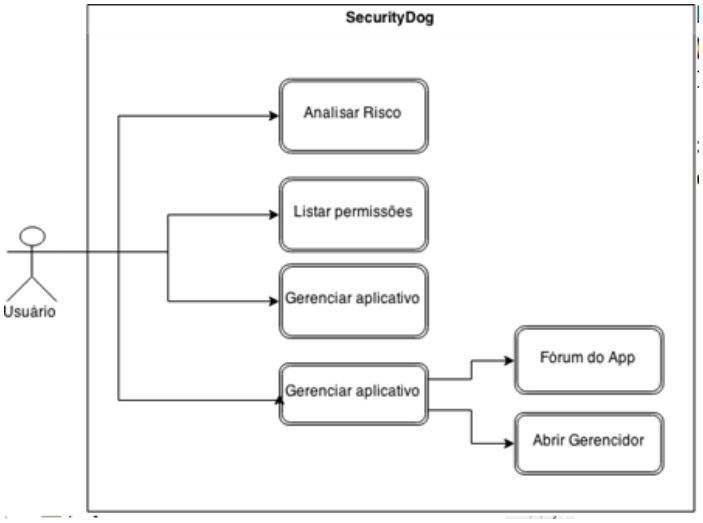
4.3 Implementação

Após os testes do modelo proposto, se fez necessário automatizá-lo, a fim de facilitar a análise de todos os aplicativos instalados no dispositivo Android. O programa foi implementado tendo como meta três princípios: Usabilidade, Confiabilidade e Eficiência (PECHANSKY, 2011), vislumbrando a premissa básica de que um sistema não intuitivo para o usuário, possivelmente terá um baixo nível de aceitação. Para facilitar a usabilidade do sistema o aplicativo foi projetado de forma a minimizar o caminho percorrido pelo usuário para executar uma operação.

Logo, para analisar o risco é necessário somente um clique e então o App fornece a lista de aplicativos e riscos. Para mostrar maior confiabilidade foi extinto o requisito de acesso root no dispositivo móvel, além disso, manteve-se o princípio de pedir apenas as permissões necessárias para o funcionamento do programa. O aplicativo preza pela eficiência, para isso foi criado o modelo de análise de risco que não necessite de grande processamento.

As funcionalidades disponíveis aos usuários podem ser observadas no diagrama de casos de uso do aplicativo na Figura 30.

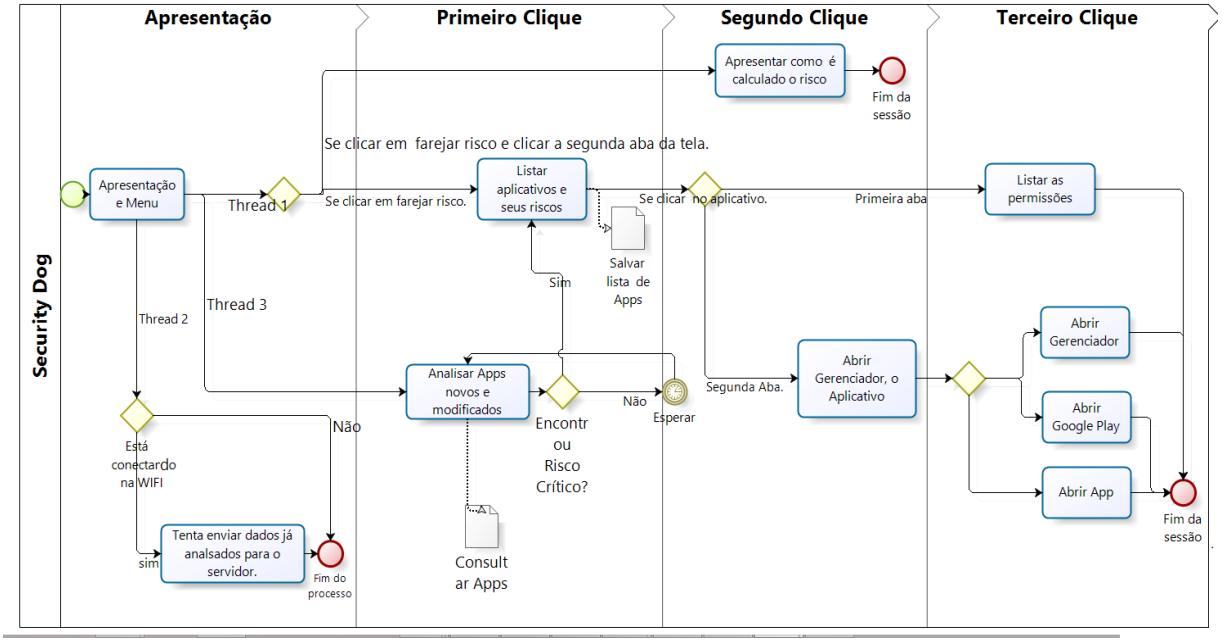
Figura 30: Diagrama de caso de uso.



Fonte: (Próprio autor, 2014).

O funcionamento do aplicativo pode ser melhor entendido com o Fluxograma da Figura 31.

Figura 31: fluxograma que representa o funcionamento do programa.



Fonte: (Próprio autor, 2014).

A figura acima ilustra todo o fluxo de funções do programa, a partir dela é possível entender todas as funcionalidades do projeto e como elas estão distribuídas; essas funções estão detalhadas nas seções seguintes distribuídas da seguinte maneira: 4.3.1 apresenta o aparato tecnológico utilizado para a construção do aplicativo, 4.3.2 mostra como é feita a apresentação do aplicativo para o usuário, 4.3.3 descreve a implementação das principais funcionalidades do Apps, 4.3.4 descreve funcionalidades extras de gerenciamento do aplicativo e, a última seção 4.3.5 apresenta o trabalho concluído no presente momento, ressalta-se, no entanto, que o mesmo poderá ser utilizado para possíveis atualizações do sistema.

4.3.1. Aparato tecnológico

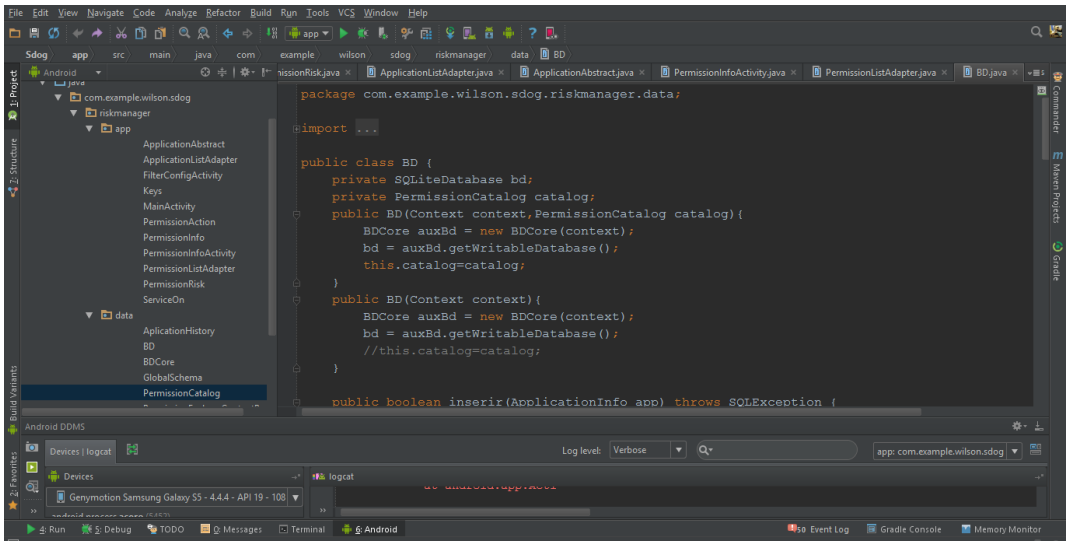
Para auxiliar a programação será utilizado um aparato tecnológico, visando facilitar a construção do aplicativo. A linguagem escolhida para a implementação foi o Java, pois é o padrão do Android, e a interface gráfica foi construída em XML, o que garante uma compatibilidade com a maioria das versões Android e demais

aparelhos a partir da versão 2.2, pois é leve, não necessitando de muita memória e processamento.

A programação Java para Android é simples, porém a instalação e configuração das ferramentas como IDE e Emuladores, se tornou um processo demorado. A IDE escolhida foi a Android Studio, que teve seu projeto financiado pela Google.

Para o ambiente de programação ficar completo foi necessária à instalação e configuração de alguns *softwares* seguindo alguns passos: Instalação do JDK 8, que pode ser baixado no site da Oracle; Instalação e configuração do Android Studio que está ilustrado na figura 32. Essa IDE foi escolhida, pois já instala e configura o SDK do Java Android automaticamente, pois em outras IDE, este processo deve ser realizado de forma manual.

Figura 32: IDE de programação Java Android.



Fonte: (Próprio autor, 2014).

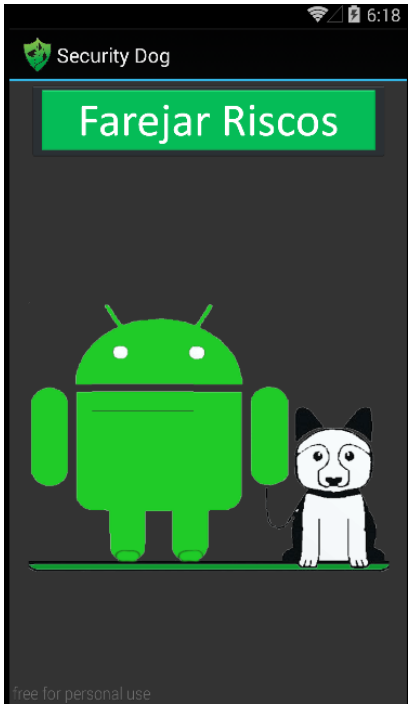
Para evitar a transferência da aplicação copilada para o dispositivo móvel e a instalação para cada teste, foi necessário instalar um emulador, que executa o App em uma máquina virtual no próprio computador. O Android Studio tem o seu próprio emulador, porém não é muito eficiente, sua inicialização a cada compilação é muito lenta, e sua usabilidade não condiz com o ambiente real de um dispositivo móvel, pois se torna mais lento, dando a impressão de falsos problemas do aplicativo ao programador.

Constatados esses problemas, optou-se pela instalação do emulador Genymotion[6]. Esse emulador trabalha criando uma máquina virtual no *virtual box*[7], com as características de dispositivos comerciais, tornando o desempenho da máquina virtual muito parecido com o da real. O Genymotion possui suporte para o Android Studion, facilitando a transição do aplicativo para a máquina virtual, logo, esse processo ocorre automaticamente ao copilar o aplicativo com o modo Debug.

4.3.2. Apresentação

Quando o programa é iniciado, a primeira tela exibida ao usuário é a de apresentação, com pouca informação, apenas com um botão, onde aparece escrito: Farejar Riscos. Ao fundo da tela, abaixo do referido botão, há uma imagem que representa o trabalho que o aplicativo irá executar no dispositivo Android, como pode ser visto na Figura 33. Dessa forma pretende-se criar uma afinidade do usuário para com o aplicativo.

Figura 33: Tela inicial do aplicativo Security Dog.



Fonte: (Próprio Autor, 2015).

[6] Gerenciador de maquinas virtuais (GENYMOTION)
[7] Cria uma emulação de um computador virtual. (VIRTUALBOX, 2015)

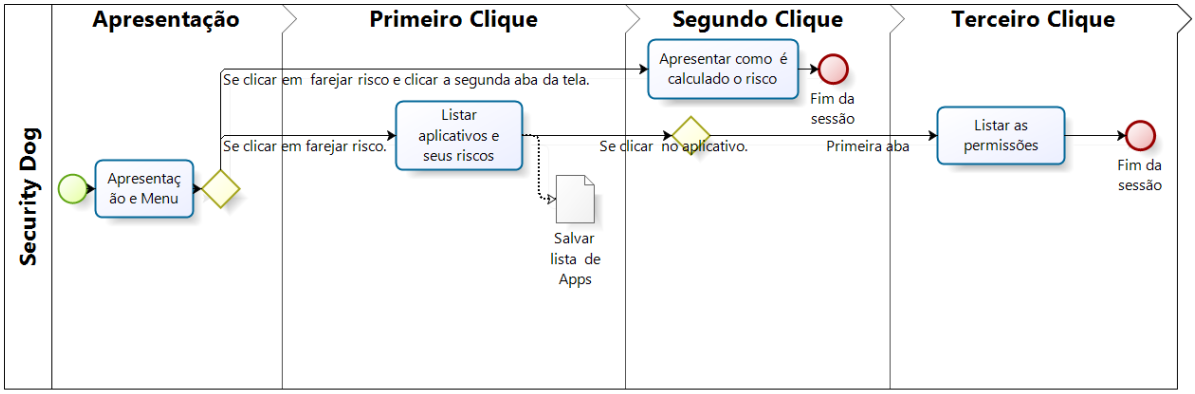
Na tela de apresentação são iniciadas 2 threads, sendo uma para usar o aplicativo em background e outra para executar o restante das funções do programa. As threads foram incluídas para executar serviços em paralelo, isso não representará perda no desempenho, já que o serviço de monitoramento não tem prioridade de execução. Este serviço de monitoramento tem que ser vinculado a uma thread secundária pois não será encerrado junto ao programa.

Uma terceira thread está sendo desenvolvida e executada nesta etapa para carregar os dados já analisados para um servidor central, mas, esta funcionalidade não está empregada na versão de teste, pois não há tempo hábil de configuração do servidor. Porém, a função será implementada, a fim de salvar no dispositivo o histórico de análises realizadas.

4.3.3. Execução Principal

O aplicativo projetado pretende informar ao usuário o risco de cada aplicativo instalado em seu dispositivo móvel, de acordo com quais permissões o mesmo solicita. O processo é executado na thread principal com maior prioridade já que está atuando diretamente com o usuário, como pode ser visualizado no diagrama da Figura 34.

Figura 34: Execução principal do programa.



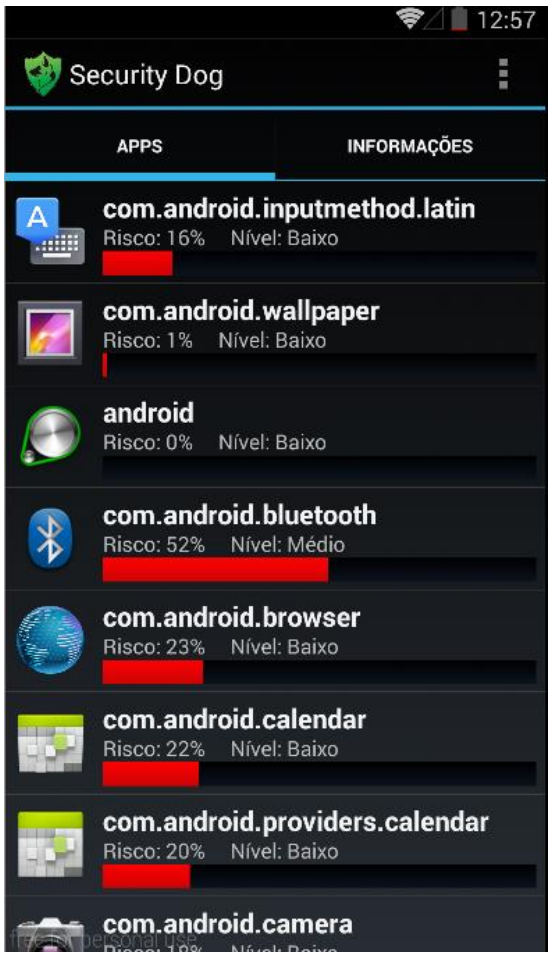
Fonte: (Próprio autor, 2014).

O processo em questão foi dividido em 3 telas: Lista de aplicativos, informações sobre cálculo de risco e tela com as permissões, todas as funcionalidades de cada tela e como foram implementadas estão descritas a seguir:

- **Lista de Aplicativos**

A Lista de aplicativos, seu risco e a explicação de como o risco é calculado, são colocados lado a lado com ajuda de um TabHost. O Risco é listado em uma *ativityList*, que está inserido na primeira aba da tela, como mostra a Figura 35.

Figura 35: Lista de aplicativos analisados.



Fonte:(Próprio autor,2014).

Para apresentar a lista de aplicativos com os seus respectivos riscos seguiu-se os seguintes passos:

- Identificação dos aplicativos instalados;
- Abertura de seu arquivo e identificação de suas permissões;

- Atribuição do risco;
- Listar na tela;
- Salvar análise no banco de dados.

Para identificação dos aplicativos instalados foi utilizada a API do Android PackageManager, que é um gerenciador de pacotes com limitações para aparelhos que não possuem acesso *root.* O seguinte código, ilustrado na Figura 36, foi utilizado para essa função :

Figura 36: Trecho do código usado para importar aplicativos instalados.

```
for (ApplicationInfo app : packManager
        .getInstalledApplications(PackageManager.GET_META_DATA))
```

Fonte: (Próprio autor, 2014).

O passo seguinte foi a identificação das permissões solicitadas pelo aplicativo. Essa função foi uma das mais trabalhosas, já que é necessário lidar com arquivos de outros aplicativos, que são de fácil acesso apenas para usuários *root*, o que não é o caso da maioria dos usuários de dispositivos móveis; contudo, resolveu-se esse problema com um aplicativo de código livre: o *Permission Explorer*. Algumas técnicas e classes foram importadas desse projeto. Respeitou-se a integridade intelectual dos autores do projeto original.

Utilizando a técnica do *Permission Explorer* foi construído o código para identificar as permissões solicitadas pelo aplicativo, conforme mostrado na Figura 37.

Figura 37: extrair as permissões de um aplicativo.

```
PackageInfo pack = packManager.getPackageInfo(
        app.packageName, PackageManager.GET_PERMISSIONS);

if (pack.requestedPermissions == null)
    return 0;

for (String permName : pack.requestedPermissions) {
```

Fonte: (Próprio autor, 2014).

Com a API *PackageInfo*, foi construída uma função para identificar as permissões de cada aplicativo e as informações do APK, como: ícone, permissões, nome e versão. Para calcular a porcentagem foi utilizado o maior risco possível e o risco atual do aplicativo. A função que retorna a porcentagem do risco é chamada getRiscoTotal e está descrita na figura 38.

Figura 38: Função getRisotoтal a classe PermissionCatalog.

```
public int getRiscoTotal(ApplicationInfo app){
    int risk=0;
    PermissionRisk permissionRisk=new PermissionRisk();

            try {
                PackageInfo pack = packManager.getPackageInfo(
                        app.packageName, PackageManager.GET_PERMISSIONS);

                if (pack.requestedPermissions == null)
                    return 0;

                for (String permName : pack.requestedPermissions) {
                      risk=permissionRisk.getRiskPermission(permName)+risk;
                }
            } catch (NameNotFoundException e) {
                throw new AssertionError();
            }

            int riskP=risk*100/permissionRisk.getRiskMax();
    return riskP;
}
```

Fonte:(Próprio autor, 2014).

A fim de facilitar uma análise intuitiva, foi inserida a avaliação qualitativa, que tem como função identificar o risco como: Baixo, Médio e Alto, como ilustra a Figura 39.

Figura 39: Trecho da função getView da classe ApplicationListAdapter.

```
if (riskP.getlow()>=riscoTotal) {
    tv.setText("Risco: " + riscoTotal + "%     Nível: Baixo");
}else
    if(riskP.getlow() < riscoTotal && riscoTotal <= riskP.getHigh()){
        tv.setText("Risco: " + riscoTotal + "%     Nível: Médio");

}else {
    tv.setText("Risco: " + riscoTotal + "%     Nível: Alto");
}
```

Fonte: (Próprio autor, 2014).

Com os aplicativos encontrados e seu risco calculado, as informações serão apresentadas para o usuário em uma lista *ListActivity,* no xml row_app como mostra a Figura 40.

.

Figura 40: Função getView da classe ApplicationListAdapter

```
protected View getView(View inflatedView, ApplicationInfo app,
        boolean justInflated) {

    PermissionRisk riskP=new PermissionRisk();

    TextView tv = (TextView) inflatedView.findViewById(R.id.app_name);
    tv.setText(app.loadLabel(catalog.getPackageManager()));

    ImageView iv = (ImageView) inflatedView.findViewById(R.id.app_icon);
    iv.setImageDrawable(app.loadIcon(catalog.getPackageManager()));


    ProgressBar pv= (ProgressBar) inflatedView.findViewById(R.id.app_bar);
    tv = (TextView) inflatedView.findViewById(R.id.app_package);

    int riscoTotal=catalog.getRiscoTotal(app);
    Drawable draw;
    pv.setProgress(riscoTotal);
```

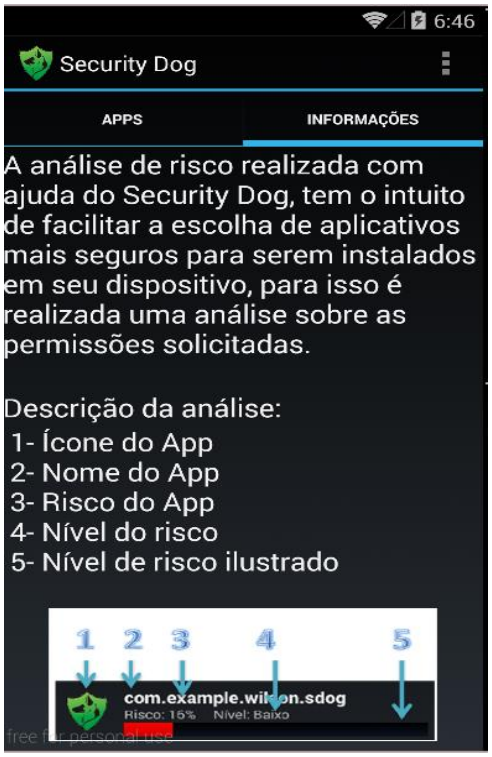
Fonte: (Próprio autor, 2014).

Após a criação da tela com os dados para o usuário, os resultados são salvos em um banco de dados SQLite nativo do Android, para que isso fosse possível, foi criada uma tabela no banco, com isso não será necessário recalcular os riscos dos aplicativos se os mesmos não sofrerem nenhuma alteração.

- **Apresentação do Cálculo de Risco**

Para um maior entendimento do usuário, o cálculo de risco será apresentado para ele em uma segunda tela com função explicativa, acessível na segunda aba do *TabHost* da tela da lista de aplicativos analisados, como ilustra a figura 41 abaixo.

Figura 41: Informações sobre o calculo de risco.



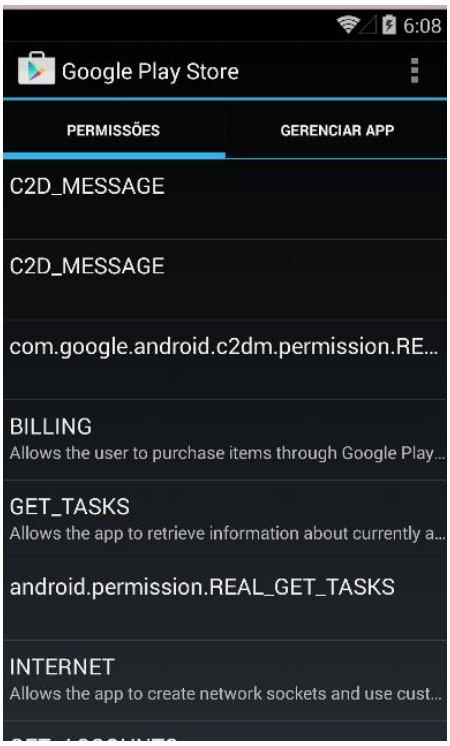
Fonte: (Próprio autor, 2014).

A descrição da análise apresenta um texto explicativo sobre o modelo e a listagem dos componentes dos aplicativos analisados, que são: Ícone, Nome, Risco, e Nível de Risco, também foram apresentados ações que devem ser tomadas para cada nível de risco identificado.

- **Listar Permissões Solicitadas**

Com o aplicativo Security Dog é possível identificar as permissões solicitadas por cada aplicativo instalado. Tal funcionalidade está presente na primeira aba da tela que é iniciada ao se selecionar um aplicativo, como ilustra a figura 42.

Figura 42: Lista de permissões solicitadas pelo aplicativo "Google Play Store".



Fonte: (Próprio autor, 2014).

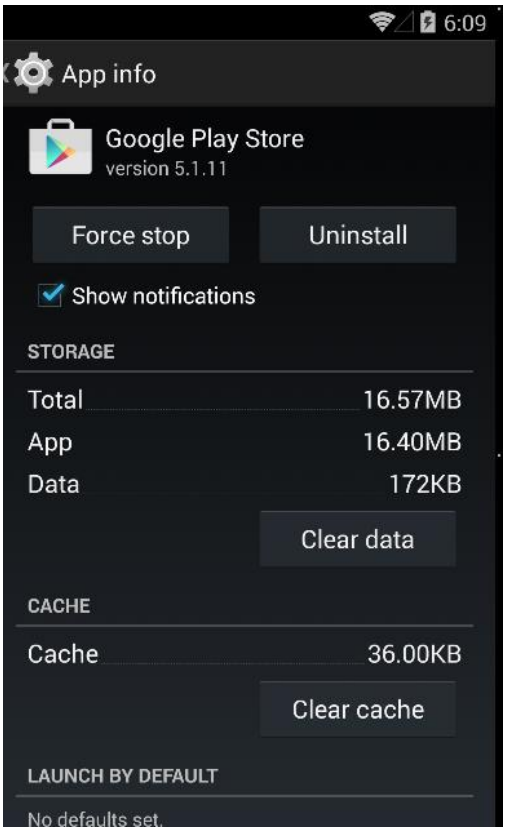
Além do nome da permissão, é apresentada uma breve descrição dos acessos que cada permissão autoriza, essa descrição é apresentada nas linguagens inglês ou português, dependendo da configuração do dispositivo.

‘ As funções construídas para criar a tela da Figura 42 encerram o processo principal de execução deste aplicativo que automatiza o modelo de análise de risco. As demais funções apresentadas tem o intuito de complementar o uso deste aplicativo.

4.3.4. Informações Adicionais

Para fornecer mais informações ao usuário sobre o aplicativo, a segunda aba da tela, ilustrada na Figura 43, conta com um link para acessar o gerenciador de aplicativos do Android, sendo possível desinstalar ou monitorar uma aplicação. Abaixo do link será aberta a página da Google Play do aplicativo, para facilitar o acesso a seu fórum.

Figura 43: Gerenciador de aplicativos do Android.



Fonte: (Próprio autor, 2014).

Para acessar o site da Google Play diretamente na página do App é necessário descobrir o nome do seu pacote ou sua assinatura. O link para o acesso a página do aplicativo deve ser “https://play.google.com/store/apps/details?id=” seguido do nome do pacote, para identificação do idioma em que a página será apresentada deve-se incluir “=pt_BR” no final do link, para o português brasileiro.

4.3.5. Serviços de Monitoramento

Para aumentar a segurança e a comodidade do usuário, este aplicativo realizará um processo de análise em segundo plano chamado de “*service”*, evitando que o usuário tenha que abri-lo periodicamente para conferir o risco de seus novos aplicativos. Esse processo é iniciado e reiniciado junto com o Android e roda em uma segunda thread que faz análises silenciosas, mesmo com o aplicativo não executado em primeiro plano. Caso a análise em segundo plano encontre algum

aplicativo com um alto risco, e que ainda não tenha sido analisado, um alerta será gerado ao usuário, esta função está ilustrada na figura 44.

Figura 44: Código da análise em que monitora a instalação de aplicativos maliciosos.

```
while(true){
    try {
        try {
            Thread.sleep(TEMPO);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        app=catalog.getScanner();
        if(app!=null){
            Intent intent = new Intent(this, ApplicationAbstract.class);
            String name=app.loadLabel(catalog.getPackageManager())+"";
            intent.putExtra(Keys.INTENT_EXT_NAME,name);
            startActivity(intent);
        }
```

Fonte: (Próprio autor, 2014).

O risco é calculado somente naqueles aplicativos que ainda não foram analisados anteriormente, para isso, é consultado o banco de dados que contém o histórico de análise. O problema desta função é a necessidade de monitoramento constante dos aplicativos instalados. Uma solução aos sistemas que possuem acesso root é monitorar os processos de instalação, e só analisar os aplicativos após uma nova instalação, mas como já descrito na sessão 2.4.6 não é possível o acesso ao monitor de processos de outros aplicativos.

4.3.6. Fornecer Estatísticas

É notável a importância desse trabalho, por essa razão deve haver um volume maior de dados, principalmente referentes às análises do aplicativo construído. Toda vez que o aplicativo for iniciado, os dados recolhidos pelo App serão carregados em um banco de dados online, essa operação pode ser complicada, pois o dispositivo nem sempre está conectado a rede, ou ainda pode estar conectado a rede 2g ou 3g, o que torna o tráfego de rede um incômodo para o usuário. Para resolver esse problema foram acrescentadas duas regras para a

exportação de dados do dispositivo: um dado não modificado só pode ser exportado um vez e, o dado só será transportado se o usuário estiver conectado em uma rede de internet WIFI.

Com essa técnica o usuário poderá contribuir com as pesquisas sem ter sua usabilidade comprometida. Nos casos em que o App já possuir conta em um banco de dados, quando for exportado, suas permissões e sua assinatura serão conferidas, caso haja alguma diferença nos dados já coletados, um alerta será emitido a fim de avisar ao usuário sobre um possível malware sendo executado em paralelo com a aplicação.

Para coletar os dados do usuário será utilizado o protocolo *Hypertext Transfer Protocol* (HTTP), o dispositivo móvel irá se conectar a uma página em PHP em um servidor central, porém não serão apresentados resultados desta funcionalidade, já que a construção do servidor não será viável para o período, contudo a funcionalidade já será parcialmente empregada ao aplicativo, necessitando apenas da configuração do servidor e atualização do sistema para o link a ser conectado.

5. VALIDAÇÃO DO APLICATIVO E ANÁLISE DE RESULTADOS

Essa seção define os planos de testes conforme o sistema Effort *Estimation*, a serem realizados durante e após a implementação do aplicativo, visando validar as funcionalidades oferecidas pelo mesmo. Serão avaliados os requisitos funcionais e de usabilidade do App.

O dispositivo móvel utilizado para os testes foi um *smartphone* LG, com sistema operacional Android 4.4.2. O aparelho possuía 13 aplicativos instalados, todos baixados da loja oficial Google Play: Calculadora, Contatos favoritos, Downloads, Facebook, Instagram, Luz do flash, Messenger, QuickMemo+, Retrica, Security Dog, Snapchat, WhatsApp e Flashlight, dentre esses apps não há nenhum malware. Além desses, será utilizado uma máquina virtual onde serão instalados 2 aplicativos: SecurityDog e Google play Store App e 2 malwares: AlSalah e Tap Snake ambos descobertos pela empresa Symante.

Os testes foram divididos em duas seções, uma descreve os testes funcionais e a outra que cita os testes de interface, realizados com os usuários.

5.1 Testes Funcionais

O teste funcional tem o objetivo de assegurar o correto funcionamento dos recursos oferecidos pelo *software* e o processamento dos dados de entrada, incluindo a navegação, fluxo dos casos de uso e resultados. O teste será repetido para as quatro principais funções: Calcular o risco, listar permissões, abrir fórum e gerenciador de aplicativo.

A primeira função testada para encontrar os aplicativos instalados no celular, calcular os seus riscos bem como os detalhes dos testes, estão descrita na Figura 45.

Figura 45: Regras para teste a função calculo de risco.

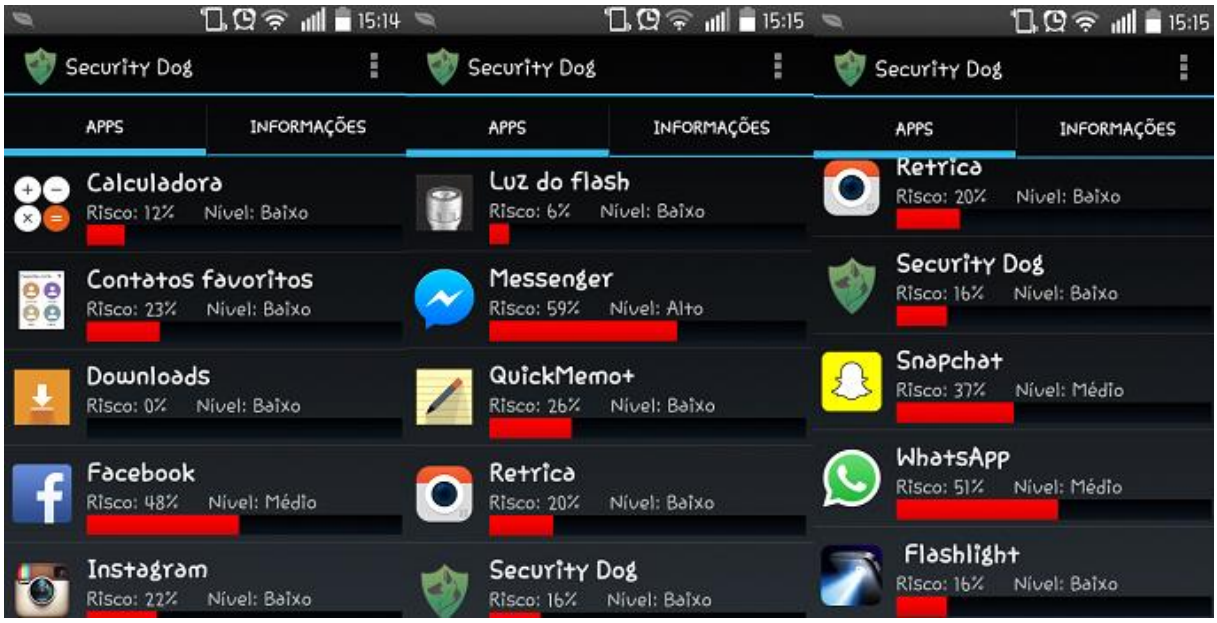
Identificador do requisito	Calcular o risco dos aplicativos instalados
Requisitos associados	Ter instalado o aplicativo Security Dog.

Entradas			**Resultados esperados**	
Nº da entrada	*Pré-condições*	*Descrição da entrada*	*Pós-condições*	*Saídas*
1	O usuário deverá ter o aplicativo instalado.	Iniciar o sistema e selecionar a opção para cálculo de risco.	O Programa irá apresentar uma lista de aplicativos e seus devidos riscos.	O Aplicativo apresenta a lista de aplicativos instalados e seus respectivos riscos.

Fonte: (Próprio autor, 2015).

A saída de dados do programa está ilustrada na Figura 46. Todos os aplicativos tiveram o risco calculado de acordo com o modelo criado, o tempo de análise é de apenas alguns segundos, não gerando grande tempo de espera ao usuário.

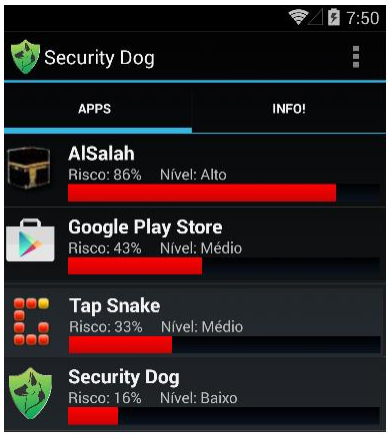
Figura 46: Teste da análise de risco do aplicativo.



Fonte: (próprio autor, 2014).

O teste também foi realizado na máquina virtual, conforme ilustrado na Figura 47.

Figura 47: Análise de risco na máquina virtual com Android.



Fonte: (Próprio autor, 2014).

Um malware foi identificado com 86% de risco, o que é alto, o outro malware teve risco médio alcançando 33%. O *software* da Google Play ficou com risco médio de 43% e o Security Dog com risco baixo de 16%. Estes resultados são favoráveis, pois o malware ALSalah foi identificado pela empresa SYMANTE (2014) como muito perigoso e o malware Tap Snake como pouco perigoso.

Em relação aos testes referentes a listagem das permissões, têm-se na figura 48 o resultado de um deles, que se refere ao aplicativo SnaPchat com 16 permissões declaradas em seu arquivo manifesto.

Figura 48: Lista permissões de um aplicativo.

Identificador do requisito	Listar permissões solicitadas por aplicativos
Requisitos associados	Ter instalado o aplicativo Security Dog.

Entradas			**Resultados esperados**	
Nº da entrada	*Pré-condições*	*Descrição da entrada*	*Pós-condições*	*Saídas*
1	Ter calculado o risco dos aplicativos.	Clicar em uma opção da lista de aplicativos analisados.	O Programa irá apresentar uma lista de permissões.	O Security Dog apresenta a lista de permissões que são solicitadas pelo aplicativo.

Fonte: (Próprio autor, 2015).

A saída de dados, ilustrada na figura abaixo, apresenta as permissões solicitadas pelo aplicativo SnaPchat, e uma breve explicação da função de cada um deles.

Figura 49: Teste de identificações de permissões do aplicativo SnaPchat.

Fonte: (Próprio autor, 2014).

Os testes para abrir o fórum do aplicativo que podem ser visualizados na Figura 50, não foram tão satisfatórios, pois o processo depende da Internet e, que o aplicativo baixado na Google Play não seja a versão mais recente, isso se justifica devido ao fato que quando o aplicativo é atualizado o nome do pacote muda, este mesmo nome é utilizado para acessar o fórum e o caminho de acesso acaba sendo perdido.

Figura 50: Abrir Fórum do Aplicativo

Identificador do requisito	Abrir fórum do aplicativo.
Requisitos associados	Ter internet e ter instalado o aplicativo Security Dog.

Entradas			Resultados esperados	
Nº da entrada	***Pré-condições***	***Descrição da entrada***	***Pós-condições***	***Saídas***
1	Ter Internet e já ter calculado o risco dos aplicativos instalados.	Clicar em uma opção da lista de aplicativos analisados e selecionar a segunda aba.	Apresentar o fórum do aplicativo.	O Programa irá apresentar a página da Google play que contem o fórum oficial do aplicativo.

Fonte: (Próprio autor, 2015).

Um dos testes realizados para abrir o gerenciador do aplicativo está descrito na Figura 51 e foi realizado no App *Snapchat*.

Figura 51: Abrir Gerenciador do aplicativo.

Identificador do requisito	Abrir fórum do aplicativo.
Requisitos associados	Ter instalado o aplicativo Security Dog.

Entradas			**Resultados esperados**	
Nº da entrada	*Pré-condições*	*Descrição da entrada*	*Pós-condições*	*Saídas*
1	Já ter calculado o risco do aplicativo.	Clicar em uma opção da lista de aplicativos analisados, selecionar a segunda aba e em gerenciador de aplicativos.	Apresentar funções de gerenciamento do aplicativo.	O Programa irá abrir o gerente de aplicativos do Android.

Fonte: (Próprio autor, 2015).

A saída dos testes foram corretas e podem ser visualizadas na Figura 52.

Figura 52: Gerenciador de aplicativos.

Fonte: (Próprio autor, 2015).

Após a realização dos testes funcionais foi possível observar que todo o sistema funciona corretamente, com exceção do fórum que possui limitações. Para solucionar este problema será proposto, em longo prazo, uma correção para o link. Na primeira versão, a tela com o link direto ao fórum será alterada para um link até a Google Play.

5.2. Testes de interface

O teste de interface tem como objetivo garantir maior e melhor interação do usuário com o aplicativo, visando garantir um sistema interativo e que opere de maneira eficiente, sem necessidade de preparo prévio por parte do usuário. A avaliação de usabilidade tem como objetivos gerais:

- Validar a eficácia da interação humano-computador, com a realização das tarefas por parte dos usuários.

- Verificar a eficiência desta interação em relação a tempo, quantidade de incidentes, passos desnecessários, busca de ajuda para realizar alguma função.
- Obter indícios da satisfação ou insatisfação do uso do sistema por parte dos usuários.

Para completar os objetivos do teste de usabilidade, as funções testadas anteriormente na avaliação de funcionalidade serão repetidas por usuários que não tiveram contato com o *software*. Foram repassados os objetivos de cada função e foram avaliados os seguintes tópicos:

- Número de erros para usar cada função;
- Número de buscas de ajuda, por função;
- Se o resultado obtido condiz com o esperado.

O indicio de satisfação em relação ao uso do sistema foi avaliado com um questionário respondido pelos usuários, eles atribuiram uma nota de 0 a 5 para três tópicos: facilidade em alcançar os objetivos, satisfação em utilizar o programa e confiança nos resultados.

O teste de usabilidade foi realizado com 3 usuários os três receberam 3 tarefas:

a- Listar o risco de todos os aplicativos instalados;
b- Listar as permissões de um aplicativo instalado;
c- Desinstalar um aplicativo pelo App.

Os dados recolhidos com os testes estão explícitos na Tabela 3 apresentada abaixo:

Tabela 3: Resultados dos testes de usabilidade.

Resultados dos Testes de Usabilidade			
	Tarefa A	Tarefa B	Tarefa C
Média de Cliques para realizar função	1	2	3.33
Número de buscas de ajuda	0	0	0
Médias notas dadas pelos usuários			
Facilidade em alcançar os objetivos	5	5	4
Satisfação em utilizar o programa	4	4,33	4
Confiança nos resultados	3.75	4,75	5

Fonte: (Próprio autor, 2015).

De acordo com gráfico acima é possível observar que o primeiro e o segundo objetivo foram concluídos com número mínimo de cliques e nenhuma ajuda externa foi necessária.

O Terceiro teste também foi concluído com sucesso. A média de 0.33 cliques a mais que o mínimo necessário, se deu ao fato dos recursos para gerência do aplicativo serem importados do próprio sistema, isso levou alguns segundos e dava sensação ao usuário que o sistema não estava respondendo, esse processo foi corrigido avisando ao usuário para esperar, pois o processo estava em andamento.

As notas dadas pelos usuários para cada função do sistema foram satisfatórias mas, isso não indica que o *software* não necessita aprimoramento, a menor média das notas foi sobre a confiança sobre a análise de risco com o valor de 3.75, mesmo assim considera-se satisfatória por se tratar de um *software* novo com nenhuma credibilidade no mercado de aplicativos. Para conquistar maior confiança dos usuários será necessário maior tempo de mercado.

6. CONSIDERAÇÕES FINAIS

O trabalho apresentou a análise histórica dos dispositivos móveis e a estatística de uso de sistemas operacionais que dominam o mercado dos dispositivos modernos. Além disso, foi apresentado um estudo sobre o Android e suas vulnerabilidades. Posteriormente, na análise dos dispositivos móveis identificou-se a importância dos SO's para tais. Entre esses, o Android tem o maior número de usuários e possui grandes vulnerabilidades que podem ser resolvidas com a identificação dos aplicativos maliciosos. Para isso, o trabalho apresentou um modelo de análise de risco e a sua automatização em um aplicativo que ajudará na tomada de decisão ao instalar ou não um determinado aplicativo, dessa forma, pode-se evitar programas que comprometam dados e usabilidade.

O modelo de análise de risco construído ao longo do trabalho apresentou bons resultados nos testes realizados, pois identificou todos os *malware*s com um risco maior que os aplicativos não infectados, apresentou boa eficiência para o cenário em que vai ser empregado, porque não tem cálculos complexos. O modelo tem grande relevância, por apresentar os dados de forma simples e clara para o usuário, sem a necessidade de um estudo prévio sobre os riscos.

O aplicativo que implementa o modelo de análise de risco criado foi concluído com êxito, mantendo bons aspectos de usabilidade e eficiência, comprovados por testes. Porém, alguns problemas foram encontrados durante esse processo, pois o Android não dá o devido acesso aos recursos do sistema. Para confrontar esses problemas, foi necessário colocar em prática conceitos aprendidos durante todo o curso.

Além da análise de riscos, o aplicativo apresenta outras funcionalidades, que podem auxiliar o usuário a gerenciar seus aplicativos. Durante a realização dos testes da ferramenta e diante de seus resultados, ficou clara a importância da pesquisa e de sua continuação, pois um bom modelo de análise de risco deve estar sempre em evolução. Para que a constante adequação e evolução do aplicativo ocorram, foram pensadas técnicas para recolher os dados das análises com o objetivo de melhorar a qualidade do modelo de análise de riscos.

Com a validação do aplicativo comprovou-se a primeira hipótese criada, onde se afirmou que seria possível construir um aplicativo para análise de risco em

dispositivos moveis. Portanto os objetivos traçados nesse trabalho foram alcançados.

REFERÊNCIAS

ANDROID STUDIO. Disponível em http://developer.android.com/sdk/index.html. Acessado em junho de 2014.

ANDROID. História. Disponível em: www.android.com/history/, acesso em dezembro de 2014.

ANDROID. **O sistema mais popular do mundo**. Disponível em: <http://www.android.com/phones-and-tablets/>, acesso em: janeiro.

BABOO. **Samsung anuncia memória 8GB LPDDR4 para dispositivos móveis.** Disponível em: <http://www.baboo.com.br/hardware/samsung-anuncia-memoria-8gb-lpddr4/> Acesso em: 15 jan.2015

BARROS, Aidil Jesus da Silveira; LEHFELD, Neide Aparecida de Souza. **Fundamentos de metodologia científica**. 3ª edição. São Paulo: Pearson PrenticeLehfeld, 2007.

BIDDLE, R; OORSCHOT, Pc Van; SOBEY, Patrick J; WHALEN, T. **Interfaces do navegador e estendeu os Certificados SSL de validação: Um Estudo Empírico**. Proc. ACM Workshop de Cloud Computing Segurança, pp 19-30, 2009.

BRAGA, Alexandre Melo; NASCIMENTO, Erick Nogueira do; PALMA, Lucas. **Introdução à Segurança de Dispositivos Móveis Modernos – Um Estudo de Caso em Android.** Minicursos do XII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais, 2012.

BURNETTE, E. Hello, Android: **Introducing Google's Mobile Development Platform.** 2. ed. USA: The Pragmatic Bookshelf October, 2009.

CHIN, E; FELTRO, AP; SEKAR, V; WAGNER, D. **Medir a confiança do usuário em Smartphone Segurança e Privacidade**. Proc. Oitava Symp. Utilizável Privacidade e Segurança. pp 1-16, 2012.

CIBRÃO, Daniel; GONSALVES, Rui. Segurança no Android. Disponivel em http://web.fe.up.pt/~jmcruz/ssi/ssi.1112/trabs-als/final/G4T10-android-final.pdf. Acessado em novembro de 2014.

CISCO. **Qual é a diferença: Vírus, worms, trojans e boots.** Disponível em:<http://www.cisco.comlweb/about/security/intelligence/virus-worm-diffs.html> Acesso em: 10 jun.2014, 16:30:30.

CORTEX-A SÉRIES. Disponível em: www.arm.com/products/processors/cortex-a/index.php Acesso em: Dezembro de 2014.

DALGLISH, B. **Tecnologia de Telecom.** Disponível em: <http://www.telecomtechstocks.com/nontechies.htm>, acessado em 23 de dezembro de 2013.

DELAC, G; SILIC, M; KROLO, J. **Emerging security threats for mobile platforms.**Proc. 34th International Convention MIPRO, pp. 1468-1473, 2011.

DESENVOLPER. Debugging. Disponível em http://developer.android.com/tools/ debugging/index.html. Acessado em novembro de 2014.

DIEDERICH, A; HEALY, Jr Busemeyer Af; PROCTOR, Rw. **Psicologia Experimental.**pp 295-319, John Wiley & Sons, 2013.

ENCK, W.; OCTEAU, D., MCDANIEL. ***A Study of Android Application Security****, Proceedings of the 20th USENIX Security Symposium, 2011.*

FEDLER, R.; KULICKE, M.; SCHUTTE, J. ***An antivirus API for Android malware recognition.*** *In: Malicious and Unwanted Software: "The Americas" (MALWARE), 2013 8th International Conference on, Issue Date: 22-24 Oct. 2013.*

FELT, AP; HA, E; EGELMAN, S; HANEY, A; CHIN, E; WAGNER, D. **Permissões do Android: a atenção do usuário, compreensão e comportamento**. Proc. Oitava Symp. Utilizável de Privacidade e Segurança, 2012.

FINUCANE, Ml; ALHAKAMI, A; SLOVIC, P; JOHNSON, Sm. **O Afeto Heurística em Acórdãos Riscos e Benefícios**. J. Tomada de Decisão Comportamental , vol. 13, n°. 1, pp.1 -17 2000.

GATES, *Christopher; CHEN, Jing; LI Ninghui.* ***Effective Risk Communication for Android Apps****. in: Dependable and Secure Computing, IEEE Transactions on, Issue Date: May-June 2014,*

GENYMOTION. Disponível em : https://www.genymotion.com. Acessado em: janeiro de 2015.

Genymotion. Disponível em https://www.genymotion.com. Acessado em Junho de 2014.

GIL, Antonio Carlos. **Como elaborar projetos de pesquisa.** 5. ed. São Paulo: Atlas, 2010.

GLOBO. **Conheça a evolução do telefone celular**. Disponível em: <http://infograficos.oglobo.globo.com/tecnologia/conheca-a-evolucao-do-telefone-celular.html> acesso em: 29 jul.2014, 17:30:10.

GOOGLE PLAY. Disponível em: <play.google.com/store> Acesso em: Outubro de 2014.

GOOGLE PLAY. ***Permission Explorer*****.** Disponível em https://play.google.com/store/apps/details?id=com.carlocriniti.android.permission_explorer&hl=pt_BR. Acessado em dezembro de 2014.

SYMATEC. Disponível em http://www.symantec.com/ security_response /landing/spam/. Acessado em janeiro de 2014.

HUANG, Chin-Yu. Analysis of a *software* reliability ***growth model with logistic testing-effort function.*** *In Software Reliability Engineering, Date Nov 1997.*

HWANG, Chang-gyu; Memory Div., Samsung Electron. Co. Ltd.. **South Korea Nanotechnology enables a new memory growth model**. In: Proceedings of the IEEE. Issue Date: Nov. 2003.

IBM. **Entendendo segurança no Android**. Disponível em http://www.ibm.com/developerworks/br/library/x-androidsecurity/. Acessado em novembro de 2014.

ITUNES. Disponível em: www.apple.com/br/itunes/. Acesso em: Outubro de 2014.

ITUNES. **Procurando por apps do IOS**. Disponível em https://itunes.apple.com/br /genre/ios. Acessado em dezembro de 2014.

KAHNEMAN, D; FAST, Pensamento; SLOW. **Farrar, Straus and Giroux**, 2011.

Kingston. Disponível em : www.kingston.com/br/flash. Acessado em: janeiro de 2015.

KOU, Xiaoming; WEN, Qiaoyan. **Intrusion detection model based on Android** in: Broadband Network and Multimedia Technology (IC-BNMT), 2011 4th IEEE International Conference.

LECHETA, Ricardo R. Google Android. **Aprenda a criar aplicações para dispositivos móveis com Android SDK**. 3ª ed. Novatec, 2013.

LEUNG, Antypas: **Improving returns on m-commerce investments,** The Journal of Business Strategy, 22 (5) (2001), pp. 12–13

MEDNIEKS, Zigurd; DORNIN, Laird; MEIKE, G. Blake; NAKAMURA, Masumi. **Programando o Android. 2ª ed**. Novatec, 2013.

MUNIZ, Joseph; LAKHANI, Aamir. ***Web Penetration Testing with Kali Linux****, Packt Publishing*, 2013.

O QUE É IOS . Disponível em: <www.apple.com/br/ios/what-is> Acesso em: Dezembro de 2014.

ORACLE. ***Download*** **do JDK**. Disponível em http://www.oracle.com/technetwork /pt/java/javase/downloads/index.html. Acessado em Junho de 2014.

PECHANSKY, Rubem**. Um modelo baseado em princípios de usabilidade para aplicação em interfaces de usuário para a interação humano-computador**. Dissertação submetida a Universidade Federal do Rio Grande do Sul, No ano de 2011.

PENG, Sancheng; SHUI, Yu; YANG, Aimin. **Smartphone *Malware* and Its Propagation Modeling. In:** Communications Surveys & Tutorials. IEEE, 2014.

PORTÕES, Christopher S; LI, Ninghui; PENG, Hao; SARMA, Bhaskar; QI, Yuan; POTHARAJU, Rahul; NITA-ROTARU, Cristina; MOLLOY, Ian. ***Generating Summary Risk Scores for Mobile Applications.*** *In: Communications Surveys & Tutorials. IEEE*, 2014.

PROCESSOR LICENSEES. Disponível em: www.arm.com/products/processors/licensees.php Acesso em: Dezembro de 2014.

PROJETO ANDROID OPEN SOURCE. **Android Visão geral de segurança**. Disponível em http://source.android.com/tech/security/. Acessado em outubro de 2014.

REDZUAN, F.; HASSIM, N. **Usability study on Integrated Computer Management System for Royal Malaysian Air Force (RMAF)**. In: e-Learning, e-Management and e-Services (IC3e), 2013 IEEE Conference on.

SCHULTZ, EE K; PL Vu; RW Proctor. **Manual de Fatores Humanos em Web Design.** pp 663-677, 2011, CRC Press.

SCHWARZ, J; MORRIS, M. **Páginas da Web aumentando e resultados de pesquisa para apoiar a avaliação Credibilidade**. Proc. SIGCHI Conf. Fatores Humanos em Sistemas de Computação, pp 1245-1254, 2011.

SNORT. Disponível em: https://www.snort.org/. Acessado em: janeiro de 2015.

SUN, J; AHLUWALIA, P; KOONG, Ks. **O mais seguro o melhor? Um Estudo de Informações de Segurança de prontidão**. Gestão Industrial e Sistemas de Dados, vol. 111, não. 4, pp 570-588, 2011

SUPPORT GOOGLE. **Rever as autorizações da Aplicação**. Disponível em: <support.google.com/googleplay/answer/6014972?p=app_permissions&rd=1> Acesso em: Dezembro de 2014.

SYMATEC. **Lista de malwares para android**. Disponível em http://www.symantec .com/security_response/landing/spam/. Acesado em julho de 2014.

SYMBIAN FOUNDATION. **Symbian project.** Disponível em http://licensing.symbian.org/. Acesado em dezembro de 2014.

TONIN, Graziela Simone. **Tendências em computação móvel**. 3 p. Universidade de São Paulo – USP. São Paulo, 2012. Disponível em: <grenoble.ime.usp.br/~gold/cursos/2012/movel/mono-1st/2305-1_Graziela.pdf>. Acesso em: 20 mar. 2013.

VERGARA, Sylvia Constant. **Projetos e relatórios de pesquisa em administração**. São Paulo: Atlas, 2006.

VIRTUALBOX. Disponível em: https://www.virtualbox.org/. Acessado em: Janeiro de 2015.

WANG, Wei; WANG, Xing; DAWEI, Feng; JIQIANG, Liu; ZHEN, Han; XIANGLIANG, Zhang. **Exploring Permission-Induced Risk in Android Applications for Malicious Application Detection** In: Information Forensics and Security, IEEE Transactions on, Issue Date: Nov. 2014.

XU, Wei; FANGFANG, Zhang; SENCUN, ZhuIn. Permlyzer: **Analyzing permission usage**. in Android applications Software Reliability Engineering (ISSRE), 2013 IEEE 24th International.

ZANONI, Felipe Sanches**. Avaliação de Arquitetura Paralela para Smartphones e Tablets utilizando GNU/Linux e MPI para Processamento de Dados**. Dissertação submetida a Escola de Engenharia de São Carlos da Universidade de São Paulo, São Paulo, 2013.