Avaliação e Recomendação de Aplicativos para Dispositivos Móveis com Foco em Segurança e Privacidade

Thiago Rocha[1], Eduardo Souto[2]

[1]FPF Tech – Manaus – AM – Brasil

[2]Instituto de Computação – Universidade Federal do Amazonas (UFAM) Manaus, AM – Brasil

thiago.rocha@fpf.br, esouto@icomp.ufam.edu.br

Abstract. *The growth in the number of mobile devices and the diversity of applications awakened the interest of malicious developers. As a result, users are scared of installing apps because of the risks associated with security and privacy since most applications require sensitive information. Recommendation systems are currently being used to choose apps. However, most approaches do not evaluate security and privacy, and when they do, only permissions are considered. In this context, this work presents a system that evaluates apps and suggests only secure applications to be installed, raising apps confiability. Preliminary experiments shows satisfactory results compared to works from the literature.*

Resumo. O crescimento no número de dispositivos móveis e a diversidade de *apps* despertaram o interesse de desenvolvedores maliciosos. Como consequência, usuários têm receio de instalar apps por causa dos riscos associados a segurança e privacidade. Atualmente, sistemas de recomendação vêm sendo utilizados para a escolha de apps. No entanto, a maioria das abordagens não avalia segurança e quando o faz leva em consideração apenas as permissões das apps. Nesse contexto, este trabalho apresenta um sistema que avalia as apps e sugere apenas aplicativos seguros para serem instalados, aumentando a confiabilidade das apps baixadas. Experimentos preliminares mostram resultados satisfatórios em comparação com trabalhos da literatura.

1. Introdução

O uso de dispositivos móveis vem aumentando desde a última década e mudando a forma como os usuários executam suas tarefas diárias e fazem negócios. Existem atualmente mais de 2 bilhões de usuários de *smartphones* em todo o mundo e espera-se chegar a 2.9 bilhões até 2020 [Statista 2019].

Como resultado desse crescimento, o número de aplicativos (ou *apps*) desenvolvidos para ajudar os usuários a executar uma grande variedade de tarefas também vem crescendo. Em 2018, somente a loja oficial de *apps* focada no sistema operacional Android (*Google Play*) já registrava 2,6 milhões de aplicativos [Mansoor 2019] e a previsão do número de downloads de aplicativos pode chegar a 258.2 bilhões até 2020 [Statista 2019].

Esses aplicativos são usados diariamente e fornecem uma variedade de serviços como chamadas telefônicas, envio de e-mail, serviço de navegação, transações bancárias, interações sociais, entre outros. Devido a grande quantidade de *apps*, sistemas de

recomendação têm sido utilizados pelos usuários para encontrar aplicativos conforme as suas necessidades e interesses. Por exemplo, a loja oficial da Google recomenda aplicativos semelhantes ao aplicativo desejado pelo usuário (aplicativo alvo) levando em consideração alguns aspectos como a categoria do aplicativo, quantidade de downloads e nome dos desenvolvedores. Entretanto, pouca atenção tem sido dada a aspectos de segurança e de privacidade pelos sistemas de recomendação, tanto de lojas de aplicativos oficiais como de sistemas de recomendação em geral [Xu et al. 2018].

Como na maioria das vezes os aplicativos requerem acesso a informações sensíveis do usuário como credenciais de contas (e.g. logins e senhas), informações sobre localização (e.g. latitude, longitude, endereços de entrega de compras) e dados financeiros (e.g. número do cartão de crédito, data de expiração, código de segurança), estes tornam se alvos em potencial de desenvolvedores de aplicativos maliciosos. Segundo a IDC [IDC 2019], a plataforma Android da Google possui 86% do mercado mundial de smartphones e tornou-se o alvo preferencial para os desenvolvedores de *apps* maliciosas.

Na literatura existem alguns estudos recentes sobre aspectos de segurança e privacidade em sistemas de recomendação. Esses trabalhos podem ser divididos em abordagens que recomendam configurações de permissões das *apps* [Rashidi et al. 2014], [Shukla 2017], [Liu et al. 2016] e abordagens que recomendam *apps* para o usuário decidir se realiza ou não a instalação [Su et al. 2015], [Jisha et al. 2018], [Zhu et al. 2014].

Os trabalhos do primeiro grupo compartilham uma limitação relacionada às permissões, que são características importantes para avaliar os riscos de segurança e privacidade de um aplicativo. Porém, não são suficientes para garantir que um aplicativo seja seguro [Akhuseyinoglu et al. 2016], [Martín et al. 2018]. Existem documentos e fóruns de segurança que já demonstraram vulnerabilidades e ataques que podem acontecer sem qualquer uso de permissões do Android [Kywe et al. 2016], [Cong Zheng et al. 2018]. Além disso, a maioria dos usuários não consegue entender como as permissões funcionam e para que elas servem. Tal fato cria uma lacuna entre as expectativas dos usuários e o comportamento das aplicações [Hong and Wang 2015].

Por outro lado, os trabalhos do segundo grupo compartilham uma limitação relacionada ao modo como as recomendações de aplicativos são realizadas. Tais recomendações, na maioria das vezes, utilizam como critério um conjunto de aplicativos que pertencem à mesma categoria da loja oficial. Tal fato pode não satisfazer as necessidades do usuário que está buscando por uma funcionalidade específica. Por exemplo, caso o usuário esteja interessado em um aplicativo semelhante ao *Whatsapp* não adianta o sistema de recomendação retornar o aplicativo do Firefox, apesar de ambos pertencerem à mesma categoria.

Para superar estes problemas, este trabalho propõe um sistema de recomendação de aplicativos com ciência de segurança e privacidade. O sistema possui uma camada de segurança que avalia uma aplicação e a classifica como maligna ou benigna. Assim, apenas aplicações classificadas como benignas (seguras) são consideradas na fase de recomendação. Após isso, são calculadas quatro pontuações: Funcionalidade, para garantir que apenas aplicações que possuam finalidades semelhantes sejam consideradas; Privacidade, para verificar se alguma aplicação está enviando informações confidenciais para servidores de terceiros de forma desnecessária (vazamento de dados); Popularidade, para obter as aplicações que possuem mais *downloads* na loja oficial; Usabilidade, para

obter as aplicações que possuem as melhores médias de revisões dadas pelos usuários na loja oficial.

Por fim, os aplicativos são ranqueados com base nas pontuações e listados junto com outras informações pertinentes para que os usuários entendam o comportamento das *apps* e possam decidir se desejam ou não instalar algum aplicativo.

As principais contribuições são: (1) Adição de um mecanismo de segurança, que utiliza aprendizagem de máquina, para considerar apenas aplicações seguras na fase de recomendação (2) Criação de uma pontuação de funcionalidade que possibilita a recomendação apenas das aplicações que executam funcionalidades similares ao aplicativo que está sendo avaliado; (3) Estudo e avaliação das características similares para mapear possíveis comportamentos que são capazes de ocasionar um problema de segurança ou privacidade. A partir desse mapeamento são criados predicados que representam uma hipótese que pode ser entendida pelos usuários. Por exemplo, se um aplicativo solicitar uma permissão de localização ou executar uma chamada de API de localização, a hipótese "O aplicativo acessa informações confidenciais (Localização)" é criada e relatada aos usuários. Esse processo é chamado de Mapeamento de Predicados Lógicos (MPL) e visa deixar claro para os usuários os comportamentos que determinada aplicação pode executar.

O restante desse trabalho está organizado da seguinte forma. Seção 2 detalha alguns trabalhos relacionados. Seção 3 apresenta a arquitetura do sistema criado para avaliação e recomendação de aplicativos móveis no ambiente Android. Seção 4 mostra os experimentos realizados. Finalmente, a Seção 5 conclui o artigo e expõe alguns trabalhos futuros.

2. Trabalhos Relacionados

Esta seção descreve alguns trabalhos que realizam recomendações e que, de alguma forma, levam em consideração algum aspecto de segurança.

Rashidi et al. [2014] propõem o RecDroid, com o objetivo de controlar as permissões de aplicativos Android em tempo real por meio de recomendações de configuração de permissões de usuários especialistas que usam aplicativos semelhantes. Com essa abordagem, os usuários comuns poderão tomar decisões de concessão de permissões corretas com base nas escolhas feitas pelos usuários especialistas, evitando problemas de segurança. A principal limitação do RecDroid, além do seu sistema de segurança ser focado apenas na configuração de permissões, é que sua implantação é dificultada pela necessidade de modificação do sistema operacional.

Su et al. [2015] analisam o tráfego de rede de aplicativos Android que possuem funcionalidades semelhantes para sugerir a instalação. Os 100 principais aplicativos Android de cada uma das 22 categorias mais populares (e.g. comunicação e jogos) foram baixados, executados e tiveram seu tráfego extraído. Uma análise da composição do tráfego de rede do aplicativo e uma medição do custo do tráfego é realizada. Essas informações, em conjunto com as avaliações de usuários e número de downloads são usadas para fazer recomendações de aplicativos. Uma limitação dessa abordagem está na dificuldade de garantir que o tráfego gerado para cada aplicativo seja suficiente para fazer boas recomendações. Além disso, a abordagem proposta pode identificar problemas de segurança relacionados apenas ao consumo desnecessário de tráfego de rede.

Jisha et al. [2018] propõem um sistema de recomendação que avalia a popularidade e os riscos de segurança com base nas classificações existentes (estrelas atribuídas aos aplicativos) na *play store* e nas permissões dos aplicativos. O sistema obtém as classificações dos aplicativos na *play store* para ranquear as *apps* em dois grupos (pontuações altas e baixas). As permissões são extraídas para estabelecer uma pontuação de segurança. Por fim, esses dados são integrados em uma interface que mostra os agrupamentos com a pontuação geral de classificação de cada aplicativo para que o usuário possa escolher qual aplicativo deseja instalar no dispositivo móvel. Como mencionado anteriormente, uma limitação desse trabalho é que a recomendação pode não atender às necessidades dos usuários, pois existem aplicativos pertencentes à mesma categoria que possuem objetivos diferentes.

Como observado, os trabalhos listados na literatura utilizam apenas as permissões para avaliar riscos de segurança. Isso ocorre porque as permissões podem ser facilmente recuperadas a partir da loja do aplicativo. O sistema desenvolvido neste trabalho amplia o processo de avaliação fornecendo uma análise sobre os aspectos de segurança, privacidade, funcionalidade, popularidade e usabilidade. Mais ainda, diferentemente dos sistemas existentes que recomendam apenas configurações de permissões ou se a aplicação deve ser instalada, o sistema de recomendação proposto visa sugerir uma aplicação similar que execute as mesmas funcionalidades da aplicação alvo, considerando prioritariamente os aspectos de segurança e privacidade. Por fim, o sistema realiza o mapeamento das características para listar todas as possíveis hipóteses que possam gerar problemas de vazamento de dados.

3. Sistema de Recomendação com Ciência de Segurança e Privacidade

Essa seção descreve a arquitetura do sistema proposto para avaliar e recomendar aplicações móveis em ambientes Android levando em consideração os aspectos de segurança, privacidade, funcionalidade, popularidade e usabilidade das aplicações. O sistema pode ser definido formalmente como:

DEFINIÇÃO 1. *Dado uma aplicação a de uma categoria c, e um conjunto de apps* $S = \{s\}$, *que possuem um conjunto de características da loja* $\{s_i\}$, *permissões* $\{p_i\}$, *chamadas de API estáticas* $\{m_i\}$ *e descrições* $\{d_i\}$, *o objetivo do sistema é avaliar uma aplicação utilizando métricas de segurança, privacidade, funcionalidade, popularidade, e usabilidade para montar uma lista de recomendações com aplicações do conjunto S que são da mesma categoria c e possuem funcionalidades similares baseadas nas suas descrições.*

A Figura 1 mostra uma visão geral do sistema proposto constituído pelos processos de extração de características, classificação e recomendação. A entrada de dados do sistema pode ser realizada por meio do *upload* da aplicação (arquivo *.apk*) ou fornecendo a URL do aplicativo na loja oficial.

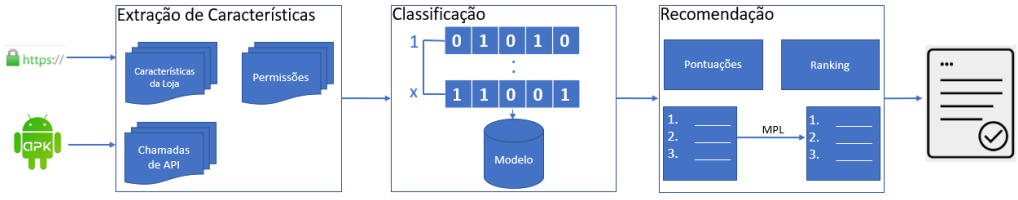

Figura 1: Visão geral do sistema proposto.

No processo de extração de características, os principais arquivos (*Android manifest.xml* e *classes.dex*) são extraídos do arquivo *.apk* (*Android Package*). Esses arquivos são utilizados para extrair as informações relacionadas às permissões e chamadas de APIs estáticas. Essas informações são agrupadas para criar um vetor de características que representa a aplicação. Os vetores de características que descrevem as aplicações são utilizados para criar um modelo de classificação (etapa de classificação) capaz de prever se uma aplicação é benigna ou maliciosa.

O processo de recomendação ocorre da seguinte forma. Caso o aplicativo que está sendo avaliado exista na *play store*, é extraída da loja uma lista de aplicativos a serem sugeridos, junto com um conjunto de características que serão usadas para avaliar os aspectos de funcionalidade e privacidade, usabilidade e popularidade da *app*. Caso o aplicativo não exista na loja, um algoritmo é usado para recuperar as descrições dos aplicativos similares que pertencem à mesma categoria do aplicativo que está sendo avaliado. Essas descrições são usadas para determinar qual descrição é mais similar à descrição do aplicativo que está sendo avaliado e, como resultado, decidir quais aplicativos serão sugeridos.

Além disso, para facilitar o entendimento do usuário, o sistema de recomendação proposto utiliza um método, denominado de mapeamento de predicados lógicos (MPL), que avalia as características para criar predicados que descrevem comportamentos que o aplicativo pode executar e os lista de uma forma que o usuário consiga entender. Por exemplo, como mostra a Figura 2 as permissões e chamadas de sistema de localização são mapeadas para o predicado "o aplicativo acessa informações confidenciais (localização)" enquanto as permissões e chamadas de sistema de envio de SMS são mapeadas para "A aplicação pode estar realizando vazamento de dados através do envio de SMS".

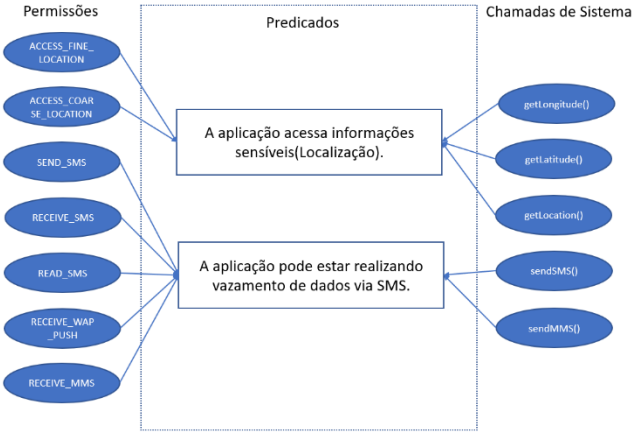

Figura 2: Exemplos de mapeamento de predicados lógicos.

Esses predicados também informam aos usuários o que os aplicativos podem enviar de informações confidencias sem autorização (vazamento de dados) e como fazem isso. As seções seguintes detalham cada etapa do sistema de recomendação.

3.1. Extração de Características

O processo de seleção e extração de característica desempenha uma tarefa essencial, pois é fundamental para garantir geração de modelos de aprendizagem eficientes. Neste artigo, com base nos estudos do sistema operacional Android e de trabalhos relacionados à recomendação e classificação de aplicativos, foram definidos três grupos de

características extraídos a partir das permissões, chamadas de APIs e características da loja oficial.

As permissões e as chamadas de API sensíveis são coletadas por meio da análise estática do arquivo *.apk* da aplicação. As chamadas de API sensíveis são obtidas a partir das *classes .dex,* as permissões são extraídas do arquivo *androidmanifest.xml* e as características da loja são extraídas por meio de um *crawler* que visita a loja oficial. No total 7 características são extraídas da loja oficial (nome do desenvolvedor, descrição, categoria, versão do Android, tamanho, número de downloads e avaliação).

As chamadas de API sensíveis são caracterizadas por métodos que podem levar ao acesso de informações privadas do usuário ou a execução de tarefas perigosas. Por exemplo, os métodos *getLatitude()* e *getLongitude()* podem ser considerados sensíveis por retornarem a localização do usuário. Neste trabalho, a seleção das APIs sensíveis é realizada com base em um mecanismo que seleciona os métodos de aplicação que executam uma ou mais operações sensíveis como acesso e envio de informação privada, recebimento e envio de tráfego de rede, operações em arquivos, envio de SMS, entre outras.

Outras características estáticas são extraídas a partir das permissões de sistema. O propósito das permissões é proteger a privacidade de um usuário Android [Android 2018]. No atual modelo de gerenciamento de permissões do Android, para ter acesso aos dados sensíveis (e.g. lista de contatos e acesso a diretórios), bem como certos recursos de sistema (como a câmera e acesso à Internet) os aplicativos precisam requisitar permissões ao usuário. Dependendo do recurso, o sistema Android pode: 1) conceder a permissão automaticamente; ou 2) exibir uma pergunta ao usuário, pedindo que ele aprove ou negue a solicitação.

No processo de extração de característica do arquivo de manifesto são considerados dois níveis de proteção definidos pelo sistema Android [Android 2018]: permissões normais e perigosas. Se as permissões forem classificadas como “normais” – ou seja, não colocam em risco a privacidade do usuário ou a operação normal do aparelho – o sistema automaticamente concede as permissões ao aplicativo. Como exemplo de permissões normais, temos o acesso ao estado das redes Wi-Fi, Bluetooth, Internet, alteração de papel de parede e vibrar. Por outro lado, se a permissão for considerada “perigosa” o usuário deve explicitamente aprovar o pedido para a aplicação. No total foram definidas 36 chamadas de API sensíveis e 60 permissões a partir da análise de 1954 aplicações que foram obtidas de [Lashkari et al. 2018]. Algumas características selecionadas também foram utilizadas em outros trabalhos relacionados à detecção de malware em Android [Yuan et al. 2016], [Feldman et al. 2015], [Ma et al. 2016].

3.2. Modelo de Classificação

O modelo de classificação é o principal responsável pela avaliação de segurança feita nas aplicações. Diferente dos trabalhos de recomendação anteriores que não utilizam aprendizagem de máquina e avaliam apenas as permissões, o sistema proposto nesse adiciona uma camada de segurança antes das recomendações serem realizadas para que apenas aplicativos rotulados como seguros possam ser sugeridos para os usuários.

As características obtidas na fase anterior são utilizadas para a criação do modelo de classificação. Cada aplicativo é representado como um vetor de características binário,

em que 1 significa que a característica é utilizada e 0 significa que não é utilizada. O modelo é criado por meio de um conjunto de treinamento com dados rotulados $\{F_j, v_j\}_{j=1}^{N_{tr}}$, onde $v_j \in \{0,1\}$ e F_j é um vetor que contém os valores binários das características. Portanto, o rótulo v mostra se os aplicativos são maliciosos ou benignos e os vetores F_j representam as características.

Formalmente, podemos representar uma aplicação A_i na forma de (F_{ai}, L_{ai}) onde F_{ai} é um vetor binário de características da aplicação A_i e L_{ai} é o rótulo da classe que pode ser (L_M, L_B) que significa que a aplicação pode ser maligna ou benigna. A Tabela 1 possui um exemplo de representação com duas aplicações (A_1 e A_2) e seus respectivos vetores de características binários compostos por cinco características. A aplicação A_1 é maliciosa (L_m) e possui três características (F_1,F_3,F_5). Portanto, seu vetor de características é <1,0,1,0,1>, enquanto que a aplicação A_2 é benigna (L_b) e possui apenas duas características e o seu vetor de características é <0,0,1,1,0>.

Tabela 1. Representação das aplicações.

Aplicação	*Características*	*Vetor de Características*	*Classe*
A_1	F_1,F_3,F_5	<1,0,1,0,1>	L_m
A_2	F_3,F_4	<0,0,1,1,0>	L_b

De posse dos vetores de características, qualquer algoritmo de aprendizagem de máquina pode ser utilizado para gerar o modelo de classificação. Os trabalhos do estado da arte que focam exclusivamente em classificação de *apps* maliciosas utilizam desde os algoritmos rasos tradicionais como o *Naive Bayes* [Dong et al. 2016] e vetores de suporte [Goyal et al. 2016] a algoritmos de aprendizagem profunda como *Deep Belief Networks* [Yuan et al. 2016] e *Autoencoders* [Hou et al. 2016].

3.3. Modelo de Recomendação

Embora seja interessante fornecer recomendações de aplicativos mais seguros, não é uma tarefa trivial estabelecer uma classificação (*ranking*) para as aplicações que levem em conta as diferentes expectativas dos usuários quanto à funcionalidade, privacidade, usabilidade e popularidade.

Para tratar este problema, este trabalho define pontuações (aqui denominadas de *scores*) para os requisitos dos usuários que são utilizadas para gerar o *ranking* final de recomendação. Especificamente, este trabalho define *scores* para funcionalidade, privacidade, usabilidade e popularidade. Com o objetivo de recomendar apenas aplicações que executam tarefas semelhantes foi atribuído uma prioridade maior para o score de funcionalidade, seguido dos *scores* de privacidade, usabilidade e popularidade respectivamente. No entanto, quando a aplicação que está sendo avaliada não existe ou não é encontrada na loja oficial apenas os *scores* de funcionalidade e privacidade são calculados.

O *score* de popularidade é calculado recuperando-se da lista de aplicativos mais semelhantes a *app* que apresentar o maior número de *downloads*. Essa *app* é considerada a mais popular. A pontuação de popularidade pode ser obtida por:

$$p = \frac{Nda * 100}{Ndp} \qquad \text{(Equação 1)}$$

onde, *p* é a popularidade do aplicativo que está sendo avaliado. N_{da} é o número de downloads do aplicativo sendo avaliado e N_{dp} é o número de downloads do aplicativo mais popular dentro da categoria das *apps* mais semelhantes.

Na loja oficial, os usuários avaliam as aplicações com base em estrelas, podendo essa avaliação ser de 1 a 5 estrelas. Assim, para auxiliar na obtenção do *score* de usabilidade é necessário calcular a média das revisões feitas pelos usuários para cada *app* dentro da loja oficial. O aplicativo com maior média é considerado o mais fácil de usar, pois geralmente as revisões são levando em consideração a *interface* gráfica da *app*. A média das revisões de uma *app* é dada por:

$$mr = \sum_{i=i}^{5} \frac{i*toti}{total} \quad \text{(Equação 2)}$$

onde, *mr* é a avaliação média das revisões de uma *app*, tot_i é o número total de votos por estrela com índice *i* e *total* é o número total de usuários que fez as revisões. Após esse cálculo, o *score* de usabilidade pode ser obtido pela equação:

$$u = \frac{Mra*100}{Mrf} \quad \text{(Equação 3)}$$

onde, *u* é a usabilidade do aplicativo que está sendo avaliado. M_{ra} é a média das revisões do aplicativo sendo avaliado e M_{rf} é a média das revisões do aplicativo mais fácil de usar.

O *score* de funcionalidade é obtido por meio de uma técnica de extração de tópicos. Um tópico é caracterizado por uma coleção de palavras que pertencem a um vocabulário fixo [Blei 2018]. Por exemplo, um aplicativo de resultados sobre partidas de futebol teria um tópico composto por palavras relacionadas às partidas como {placar, tabela, times, liga, gols}, enquanto um aplicativo da bolsa de valores teria um tópico formado por palavras relacionadas ao mercado de ações tais como{ações, mercado, corretoras}.

Para a extração de tópicos são utilizadas as descrições dos aplicativos que pertencem à mesma categoria do aplicativo que está sendo avaliado. De posse dos tópicos, uma distribuição de probabilidade é estimada. O objetivo é calcular a probabilidade de cada tópico estar inserido na descrição da *app*. Por exemplo, uma descrição pode ter uma probabilidade de 0,9 de estar relacionada a um tópico específico e 0,1 de estar relacionada a outro tópico. A soma da probabilidade será sempre 1.0. Basicamente, esta abordagem é utilizada para agrupar os aplicativos que possuem alta probabilidade de pertencerem a um tópico específico.

Esse processo é conhecido como modelagem de tópicos probabilística [Blei 2018], sendo composto por um conjunto de algoritmos estatísticos que analisam as palavras das descrições para descobrir tópicos que a compõem, como eles mudam com o tempo e como estão conectados. Neste trabalho, o algoritmo *Latent Dirichlet Allocation* (LDA) [Blei 2018] foi escolhido para executar esta tarefa.

O *score* de privacidade é calculado levando em conta apenas os aplicativos dentro da mesma categoria que possuem funcionalidades semelhantes. Neste trabalho, um aplicativo é considerado semelhante se apresentar *score* de funcionalidade superior a 70%, essa porcentagem foi escolhida por meio de testes preliminares realizados no algoritmo proposto.

Aplicativos que executam as mesmas funcionalidades tendem a exigir as mesmas permissões e, na maioria das vezes, acessam as mesmas chamadas de API, visto que executam tarefas similares. Por exemplo, dois aplicativos relacionados a serviços de localização precisam de permissões de GPS. Portanto, nesse caso uma configuração de permissão para acesso a localização não é considerada no cálculo de *score* de privacidade, pois todos os aplicativos de serviços de localização necessitam dessa configuração de permissão. Assim, para o cálculo do score de privacidade, somente as características incomuns dos aplicativos são utilizadas. A Equação 4 descreve como o score privacidade é obtido:

$$pr = \sum_{i=1}^{n}(d_i * wd) + (n_i * wn) + (c_i * wc). \qquad \text{(Equação 4)}$$

onde, *pr* é a pontuação de privacidade, d_i uma permissão perigosa com índice *i*, *wd* é o peso das permissões perigosas, n_i uma permissão normal, como classificadas em [Android 2018], com índice *i*, w*n* é o peso das permissões normais, c_i uma chamada de API com índice *i* e *wc* é o peso da chamada de API. Os pesos foram definidos levando-se em consideração o risco de cada permissão. Assim, as permissões perigosas possuem um peso maior que as permissões normais. Como mencionado na Seção 3.1, este trabalho adota o modelo de gerenciamento de permissões do Android que define quais permissões são normais e quais são perigosas. Portanto, esse *score* visa informar aos usuários o risco que a *app* possui para enviar informações confidenciais para entidades não autorizadas.

Por fim, um *ranking* final é criado com base nos *scores* de funcionalidade, privacidade, usabilidade e popularidade para mostrar aos usuários quais aplicativos são recomendados para instalação e uso. Como dito anteriormente o *score* de funcionalidade é o primeiro a ser considerado para que sejam consideradas apenas as aplicações que possuem a mesma funcionalidade, seguido pelos *score* de privacidade, usabilidade e popularidade.

Além disso, é realizado o MPL onde são agrupadas as características extraídas (permissões e chamadas de API) que são semelhantes. Em seguida são mapeados os possíveis comportamentos que podem levar a um problema de segurança ou privacidade e são criados predicados que representam uma hipótese que pode ser entendida pelo usuário. Por fim, é gerado um sumário para agrupar todas as informações obtidas como o nome da *app*, nome do desenvolvedor, resultados do MPL e o *ranking* das *apps* com os respectivos *scores*.

4. Experimentos e Resultados

Esta seção descreve os testes conduzidos para avaliar a abordagem proposta. Experimentos foram realizados e seus resultados comparados com o trabalho desenvolvido em [Jisha et al. 2018], denominado RSPSA. Este trabalho foi descrito na Seção 2 e foi escolhido por possuir características semelhantes ao sistema proposto neste trabalho.

Para realizar essa avaliação, dez aplicativos maliciosos de diferentes categorias foram selecionados e obtidos do site Koodous [Koodous 2017], que possui uma coleção de *.apks* maliciosas. Tais aplicativos foram selecionados por realizarem tarefas semelhantes às executadas pelos usuários em suas versões de aplicativos benignos. Por exemplo, o Voice SMS é um aplicativo que traduz voz em texto e envia SMS, o ClockPlus é um cronômetro para medir o tempo de uma atividade como jogos, culinária e educação,

enquanto os aplicativos Banco do Brasil e o Sberbank são aplicativos financeiros. Alguns aplicativos também são versões reempacotadas (versão modificada com código malicioso) de aplicativos conhecidos, como o Opera Mini 6.5, que é uma versão modificada do navegador Opera e Cut the Rope, que é uma versão modificada de um famoso jogo. A Tabela 2 lista os aplicativos.

Tabela 2. Aplicativos maliciosos escolhidos para teste.

Nome	*Categoria*	*Possível Comportamento*
Spam Guard	Productivity	Obtém informações confidenciais.
Cut The Rope Unlock	Puzzle	Instala ou desinstala pacotes. Faz instalação de arquivos adicionais.
Deal&Be Millionaire	Trivia	Obtém informações confidenciais.
Fish Aquarium Live Lock	Personalization	Obtém informações confidenciais.
Voice SMS	Communication	Obtém informações confidenciais e envia por SMS.
Where is My Water?	Puzzle	Obtém informações confidenciais.
ClockPlus	Tools	Envia SMS para serviços pagos.
Banco do Brasil	Finance	Obtém informações confidenciais e mostra propagandas.
Sberbank	Finance	Obtém acesso de administrador e acessa informações confidenciais.
Opera Mini 6.5	Communication	Obtém informações confidenciais.

Além das aplicações maliciosas, 1954 *apps* foram obtidas em [Lashkari et al. 2018] e utilizadas para a fase de treinamento e criação do modelo de classificação usado no protótipo do sistema de recomendação proposto. O conjunto conta com apps malignas e benignas que representam 47 categorias de aplicativos da loja oficial como esportes, produtividade, comunicação, finanças e negócios. O protótipo do sistema de recomendação proposto foi desenvolvido em linguagem de programação Java. Como não é foco deste artigo a classificação de *apps*, mas sim a recomendação de *apps* seguras, este trabalho utilizou uma árvore de decisão (J48) na etapa de classificação. Como mencionando na Seção 3.2, outros algoritmos de classificação podem ser empregados para fazer a predição de *apps* maliciosas.

O primeiro aplicativo malicioso avaliado é o Spam Guard, que pertence à categoria *productivity*. Sua descrição diz que seu objetivo é detectar e mover automaticamente os emails de SPAM da caixa de entrada para a pasta SPAM. No entanto, o aplicativo acessa informações confidenciais, como contatos e envia mensagens SMS.

A estratégia de recomendação usada no RSPSA recebe uma lista de aplicativos da mesma categoria (produtividade, neste caso) e calcula as avaliações das *apps* feitas pelos usuários e o *score* de privacidade com base nas permissões. Posteriormente, esses resultados são usados em um algoritmo de agrupamento para fazer as sugestões. A Tabela 3 mostra os três primeiros aplicativos relacionados ao Spam Guard que são sugeridos pelo RSPSA e os três aplicativos sugeridos pelo sistema proposto.

Nos resultados do RSPSA nenhum dos aplicativos da Tabela 3 executa as mesmas funcionalidades do Spam Guard. Por exemplo, Xodo PDF é um leitor e editor de arquivos no formato PDF enquanto o Business Calendar 2 é um calendário. Enquanto isso, o sistema proposto descarta da recomendação a *app* maliciosa e sugere aplicativos que possuem funcionalidades semelhantes, com destaque para o *Email Spam Filter* que é uma aplicação utilizada para restringir qual email pode ser adicionado a caixa de entrada de um usuário.

Tabela 3. Resultados com o Spam Guard.

Abordagem	*Aplicação*	*Score de Usabilidade*	*Score de Privacidade*
RSPSA	Xodo PDF Reader & Editor	4.72	7.0
	Business Calendar 2	4.61	28.0
	Password Safe - Secure Password Manager	4.61	4.0
Sistema Proposto	Email Spam Filter	2.60	11.0
	Email - Fast & Secure mail for Gmail Outlook & more	4.60	18.0
	Microsoft Outlook	4.33	16.0

Além disso, a *app Email Spam Filter* obteve *score* de usabilidade de 2.6 e um *score* de privacidade 11.0. O baixo *score* de usabilidade demonstra que a *app* tem pouca aceitação com os usuários nos quesitos facilidade de uso, interface gráfica, entre outros. No entanto, a *app* possui poucas revisões (193) e pode melhorar o seu *score* com o tempo através de novas revisões feitas pelos usuários e de atualizações no *app*. O *score* de privacidade mostra que o aplicativo não configura muitas permissões, sendo sete permissões normais, de um total de 33 e três perigosas (*READ_PHONE_STATE, WRITE_EXTERNAL_ STORAGE* e *READ_ EXTERNAL_STORAGE*), de um total de 27. A primeira configuração de permissão perigosa permite que informações sobre o dispositivo móvel sejam obtidas tais como informações de rede e o número do dispositivo, enquanto que as demais configurações de permissões possibilitam a leitura e escrita em armazenamento externo. Para efeito de comparação, o valor do *score* de privacidade não está normalizado devido ao RSPSA não ter feito nenhuma normalização. Assim, quanto maior for o *score* de privacidade, maior o risco de vazamento de dados.

A Figura 3 apresenta um *screenshot* da interface do usuário com os resultados referentes ao *Email Spam Filter* com o nome da app, categoria, desenvolvedor e *scores.* No caso do MPL foram encontrados acesso ao IMEI e escrita em armazenamento externo. Essas informações são agrupadas e passadas aos usuários para que fiquem cientes que a app pode acessar essa informação confidencial e causar um possível vazamento de dados.

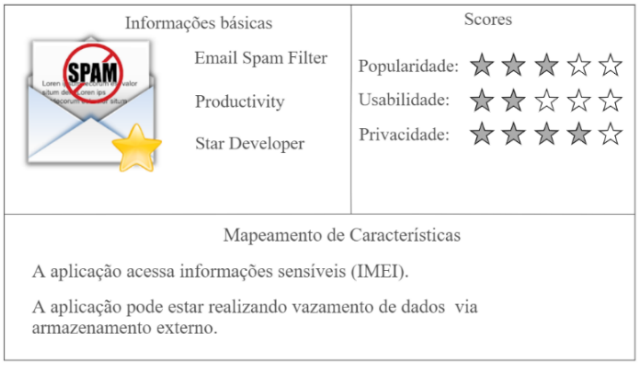

Figura 3: Interface com os resultados do *Email Spam Filter*.

A Tabela 4 mostra os resultados dos outros 9 aplicativos que foram escolhidos no protocolo experimental com as principais recomendações do RSPSA e do sistema proposto. O Banco do Brasil é um famoso banco brasileiro que possui um aplicativo móvel para pagamento de contas, transferência de dinheiro, entre outras operações bancárias. O aplicativo malicioso é uma versão reempacotada do aplicativo legítimo com o objetivo de roubar informações e mostrar propagandas. O aplicativo sugerido pelo RSPSA foi o Canadian Mortgage App enquanto que o protótipo sugeriu a versão real do

aplicativo do Banco do Brasil na loja oficial. A descrição da app maliciosa fez com que a versão real e segura fosse sugerida por causa do *score* de funcionalidade e também porque a descrição possui palavras chaves relacionadas ao Banco do Brasil como a abreviação “BB”.

Tabela 4. Resultados das Aplicações Maliciosas.

***Aplicações* Maliciosas**	***Categoria***	***RSPSA***	***Sistema Proposto***
Cut The Rope	Puzzle	I love Hue	Cut The Rope (Benigno)
Banco do Brasil	Finance	Canadian Mortgage App	Banco do Brasil (Benigno)
Deal&Be Millionaire	Trivia	Millionaire Trivia: Who Wants To Be a Millionaire?	Millionaire 2019 - Trivia Quis
Fish Aquarium Live Lock	Personalization	New Year 2019 countdown	Fish Live Wallpaper 2018
Voice SMS	Communication	Pleymojis	Write Voice SMS
Where is My Water?	Puzzle	I Love Hue	Where's My Water? 2
ClockPlus	Tools	Post	Multi Timer StopWatch
Sberbank	Finance	Canadian Mortgage App	Sberbank Mobile Bank
Opera Mini 6.5	Communication	Pleymojis	Opera Mini - fast web browser

Deal & Be Millionaire é um jogo da categoria *Trivia*. É um *malware* que obtém informações sensíveis, sua descrição corresponde a um jogo para se tornar o próximo milionário através de negócios intensos e decisões arriscadas. RSPSA recomendou o aplicativo *Millionaire Trivia: Who Wants To Be a Millionaire?*, enquanto que o protótipo sugeriu o *Millionaire 2019 - Trivia Quis*.

Este é o único caso em que o RSPSA sugeriu um aplicativo que tem a mesma funcionalidade do aplicativo que está sendo avaliado. Como o RSPSA obtém os aplicativos de uma categoria e calcula um score de pontuação dentro da categoria, ele sempre retornará o mesmo aplicativo sugerido. Por exemplo, como visto na Tabela 4 para a categoria *Communication*, o RSPSA sempre retorna Pleymojis, enquanto que para a categoria *Finance*, sempre é retornada a Canadian Mortgage App.

Por outro lado, o sistema de recomendação proposto sempre retorna resultados diferentes, visto que o mesmo leva em consideração as descrições dos aplicativos para verificar suas funcionalidades. Por exemplo, dentro da categoria Communication nos resultados da Tabela 4 foram sugeridos dois aplicativos diferentes como o Write Voice SMS: write sms by voice para o aplicativo Voice SMS e o navegador web real Opera mini para o aplicativo Opera Mini 6.5.

5. Conclusões e Trabalhos Futuros

Esse trabalho apresentou um sistema para avaliação e recomendação de aplicativos móveis no ambiente Android com ciência de segurança e privacidade. A adição de uma camada de segurança antes da execução do mecanismo de recomendação mostrou que é possível realizar a recomendação levando-se em consideração apenas aplicativos seguros. Além disso, o *score* de funcionalidade fez com que apenas aplicações que possuem a mesma finalidade fossem sugeridas e o *score* de privacidade aumentou a confiabilidade e entendimento dos usuários com relação a vazamento de dados.

Por fim, a avaliação do protótipo implementado, com 1954 aplicações benignas e maliciosas, mostrou que o modelo proposto obteve resultados satisfatórios quando

comparado com outras abordagens de recomendação. Como extensão deste trabalho pretendemos identificar novas características que possam levar a caracterização de aplicativos maliciosos, aumentar a base de aplicativos para recomendação e realizar comparações com outros trabalhos do estado da arte.

Agradecimentos. Essa pesquisa foi parcialmente patrocinada pela Samsung Eletrônica da Amazônia, dentro dos termos da lei federal brasileira nº 8.387/1991.

Referências.

Akhuseyinoglu, Nuray Baltaci, and Kamil Akhuseyinoglu. 2016. "AntiWare: An Automated Android Malware Detection Tool Based on Machine Learning Approach and Official Market Metadata." *Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*.

Android."Requesting Permissions." Disponível em: https://developer.android.com/guide/topics/permissions/requesting.html, Janeiro 2018.

Bilić, Denise Giusto."Balanço 2017: Análise de Riscos E Ameaças Para Dispositivos Móveis." Disponível em: https://www.welivesecurity.com/br/2017/12/29/balanco-2017-riscos-e-ameacas-para-dispositivos-moveis/, Abril 2017.

Cong Zheng, Wenjun Hu, Xiao Zhang, Zhi Xu. "Cloak and Dagger Attack with No Permission." Disponível em: https://unit42.paloaltonetworks.com/unit42-android-toast-overlay-attack-cloak-and-dagger-with-no-permissions/, Abril 2017.

David Blei, 2012. "Probabilistic Topic Models" *Communications of the ACM*, páginas 77-84.

Feng Dong, Yanhui Guo, Chengze Li, Guoai Xu e Fang We, 2016 "ClassifyDroid: Large scale Android applications classification using semi-supervised Multinomial Naive Bayes" *International Conference on Cloud Computing and Intelligence Systems (CCIS)*.

Haoyu Wang, Jason Hong e Yao Guo. 2015 "Using Text Mining to Infer the Purpose of Permission Use in Mobile Apps" *International Joint Conference on Pervasive and Ubiquitous Computing*, páginas 1107-118.

Hengshu Zhu, Hui Xiong, Yong Ge e Enhong Chen 2014. "Mobile App Recommendations with Security and Privacy Awareness Categories and Subject Descriptors." ACM SIGKDD *International conference on Knowledge discovery and data mining*, páginas 951-960.

Jisha, R C, Ram Krishnan, and Varun Vikraman. 2018. "User Ratings and Permissions." *International Conference on Advances in Computing, Communications and Informatics,* páginas 1000–1005.

Koodous. "Koodous." Disponível em: https://koodous.com/, Janeiro 2019.

Kywe, Su Mon, Yingjiu Li, Kunal Petal, and Michael Grace. 2016. "Attacking Android Smartphone Systems without Permissions." *Annual Conference on Privacy, Security and Trust (PST).*

Lashkari, Arash Habibi, Andi Fitriah A Kadir, Laya Taheri, and Ali A Ghorbani. "Toward Developing a Systematic Approach to Generate Benchmark Android Malware Datasets and Classification."*International Carnahan Conference on Security*

Technology (ICCST), páginas 1–7.

Li Ma, Yuexiang Yang, Xiaolei Wang e Jie He 2016. "Ultra-Lightweight Malware Detection of Android Using 2-Level Machine Learning," *International Conference on Information Science and Control Engineering (ICISCE)*, páginas 729–733.

Liu Rui, Jiannong Cao, and Kehuan Zhang. "When Privacy Meets Usability : Unobtrusive Privacy Permission Recommendation System for Mobile Apps Based on Crowdsourcing" *IEEE Transactions on Services Computing,* páginas 864-878.

Mansoor Lqbal. "App Download and Usage Statistics" Disponível em: https://www.businessofapps.com/data/app-statistics/, Abril 2019.

Martín, Ignacio, José Alberto Hernández, Alfonso Muñoz, and Antonio Guzmán. "Android Malware Characterization Using Metadata and Machine Learning Techniques" *International Journal of Security and Communication Networks*.

Rashidi, Fung and Vu. 2014. "RecDroid: A Resource Access Permission Control Portal and Recommendation Service for Smartphone Users."*ACM MobiCom Workshop on Security and Privacy in Mobile Environments*, páginas 13–17.

Rohit Goyal, Angelo Spognardi, Nicola Dragoni e Marios Argyriou, 2016 "SafeDroid: A distributed malware detection service for android," *International Conference on Service-Oriented Computing and Applications (SOCA)*, páginas 59–66.

Xin Su, Dafang Zhang, Wenjia Liy e Wenwei Li 2015 "Android App Recommendation Approach Based on Network Traffic Measurement and Analysis." *International Symposium on Computers and Communication (ISCC)*, páginas 112-118.

Shifu Hou, Aaron Saas, Lifei Chen e Yanfang Ye, 2016 "Deep4MalDroid: A deep learning framework for android malware detection based on Linux kernel system call graphs," *International Conference on Web Intelligence Workshops (WIW)*, páginas 104–111.

Shukla, Ankur. 2017. "Permission Recommender System for Android" *International Conference on Security of Information and Networks,* páginas 13–16.

Statista. "Annual Number of Mobile Apps Downloads." Disponível em: https://www.statista.com/statistics/271644/worldwide-free-and-paid-mobile-app-store-downloads/, Maio 2019.

Statista. "Number of Smartphone Users Worldwide." Disponível em: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/, Novembro 2019.

Stephen Feldman, Dillon Stadther e Bing Wa 2015. "Manilyzer: Automated Android malware detection through manifest analysis," *International Conference on Mobile Ad Hoc and Sensor Systems*, páginas 767–772.

Xu, Kun, Weidong Zhang, and Zheng Yan. 2018. "A Privacy-Preserving Mobile Application Recommender System Based on Trust Evaluation." *Journal of Computational Science*, páginas 87–107.

Zhenlong Yuan, Yongqiang Lu e Yibo Xue 2016. "Droiddetector: android malware characterization and detection using deep learning," Tsinghua Science and Technology, páginas 114–123.