Mobile App Recommendations with Security and Privacy Awareness

Hengshu Zhu[1] Hui Xiong[2*] Yong Ge[3] Enhong Chen[1*]
[1]University of Science and Technology of China, [2]Rutgers University, [3]UNC Charlotte
zhs@mail.ustc.edu.cn, hxiong@rutgers.edu, yong.ge@uncc.edu, cheneh@ustc.edu.cn

ABSTRACT

With the rapid prevalence of smart mobile devices, the number of mobile Apps available has exploded over the past few years. To facilitate the choice of mobile Apps, existing mobile App recommender systems typically recommend popular mobile Apps to mobile users. However, mobile Apps are highly varied and often poorly understood, particularly for their activities and functions related to privacy and security. Therefore, more and more mobile users are reluctant to adopt mobile Apps due to the risk of privacy invasion and other security concerns. To fill this crucial void, in this paper, we propose to develop a mobile App recommender system with privacy and security awareness. The design goal is to equip the recommender system with the functionality which allows to automatically detect and evaluate the security risk of mobile Apps. Then, the recommender system can provide App recommendations by considering both the Apps' popularity and the users' security preferences. Specifically, a mobile App can lead to security risk because insecure *data access permissions* have been implemented in this App. Therefore, we first develop the techniques to automatically detect the potential security risk for each mobile App by exploiting the requested permissions. Then, we propose a flexible approach based on *modern portfolio theory* for recommending Apps by striking a balance between the Apps' popularity and the users' security concerns, and build an *App hash tree* to efficiently recommend Apps. Finally, we evaluate our approach with extensive experiments on a large-scale data set collected from Google Play. The experimental results clearly validate the effectiveness of our approach.

Categories and Subject Descriptors

H.2.8.d [**Information Technology and Systems**]: Database Applications - Data Mining

Keywords

Mobile Apps, Recommender Systems, Security and Privacy

*Corresponding Author.

KDD'14, August 24–27, 2014, New York, NY, USA.

http://dx.doi.org/10.1145/2623330.2623705.

1. INTRODUCTION

Recent years have witnessed the rapid and increased prevalence of smart mobile devices, such as smart phones, a huge number of mobile Apps have been developed for mobile users. For example, as of the end of July 2013, the Google Play has had over 1 million Apps and there have been over 50 billion cumulative downloads, and these numbers are still growing dramatically. Due to the prospering mobile App industry, the functionalities of smart devices have been intensely extended to meet diversified user needs. However, mobile Apps are highly varied and often poorly understood, particularly for their activities and functions related to privacy and security. Indeed, to improve user experiences, more and more advanced mobile Apps are committed to provide intelligent and personalized services for users, such as location based services and social sharing services. These services usually involve access permissions of users' personal data, such as real-time locations and the contact lists.

However, such intelligent mobile Apps may result in the potential security and privacy risks for users. For instance, users may not expect their locations (e.g., home locations, workplaces) and other privacy information (e.g., contact lists, SMS records) to be spied by the third party Apps. In fact, as reported by NBC News [1], consumers have grown so concerned about privacy on their mobile phones. Many consumers have avoided downloading some mobile Apps, and many others have removed Apps which may have access to their personal data. Also, a recent survey from IDG News [2] reveals that 54% of U.S. mobile App users surveyed have decided not to install an App when they discovered how much personal information it would collect, and 30% of App users have uninstalled an App after learning about the personal information it collected. Therefore, the development of a mobile App recommender system with security and privacy awareness becomes critical for the healthy development of the mobile App industry.

In the literature, there are recent studies about security and privacy issues of mobile Apps, and mobile App recommendations. For example, some works are focused on malware code detection [6, 13], the security middleware development [7, 20], and the App access permission model development [5, 8]. However, these works either need to analyze the source code of each mobile App, or detect the system API calls during the App running. Indeed, these approaches are very hard to be implemented in practice, since it is not a trivial task to efficiently and accurately detect the malware

[1]http://www.nbcnews.com/
[2]http://www.idg.com/

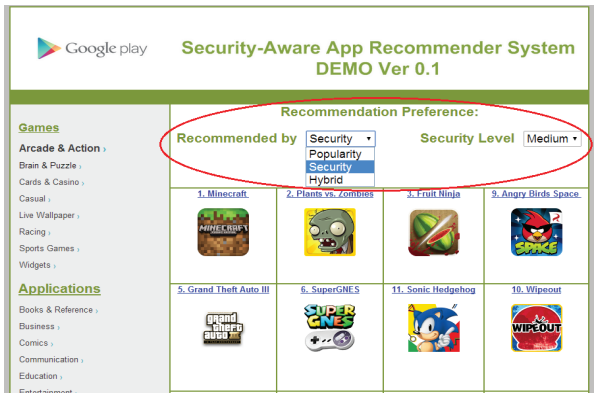


Figure 1: A demo system of mobile App recommendations with security and privacy awareness.

codes for each mobile App and users often do not want some security software to frequently scan their devices. Meanwhile, in the area of mobile App recommendation, some works studied the personalized App recommendation methods [17], the intelligent mobile App recommendations by exploiting enriched contextual information [10, 21], and the problem of App ranking fraud detection [22]. However, all these works only consider user preferences about the Apps' popularity (e.g., ratings, downloads), but not the security and privacy risks inherent in the mobile Apps.

To this end, in this paper, we propose to develop a mobile App recommender system with security and privacy awareness. The design goal is to equip the recommender system with the ability to automatically detect and evaluate the security and privacy risks of mobile Apps. Also, when applying this recommender system for App recommendations, it should be able to strike a balance between the Apps' popularity and the users' security preferences. Figure 1 shows the interface of our demo system for mobile App recommendations with security and privacy awareness. In this system, users can select different evaluation metrics, such as Popularity, Security, and Hybrid, to obtain App recommendations with respect to their preferred security levels. While we do not aim at developing personalized App recommender systems because the individual download statistics and App usage data are often not publicly available, our **non-personalized** App recommendations by considering both popularity and security are very important for mobile App services. For instance, both Apple and Google provide non-personalized top paid/free App recommendations based on the popularity information (e.g., overall download and rating) every day. However, they do not explore and consider the security preferences in their recommended top charts. Indeed, the developed system will be beneficial for the healthy development of the mobile App industry.

However, there are two critical challenge for developing an App recommender system with security and privacy awareness. Specifically, the first challenge is how to effectively identify the security risks of mobile Apps from the large-scale mobile App data. The second challenge is how to strike a balance between the Apps' popularity and the users' concerns about security and privacy. Indeed, our careful observation reveals that the potential security risks of mobile Apps are essentially caused by the *data access permissions* of each App, such as permissions requested for accessing real-time locations. Therefore, in this paper, we first propose to exploit the requested permissions for detecting the potential security risk of each mobile App. The proposed approach is based on random walk regularization with an *App-permission bipartite graph*, which can learn the security risk of mobile Apps automatically without relying on any predefined risk function. Furthermore, based on the *modern portfolio theory* [16], we develop a flexible optimization approach for recommending Apps by considering both Apps' popularity and users' concerns about security and privacy. Particularly, there are often many different security preferences of mobile users, and a huge number of Apps as candidates for recommendations. To enhance the performances of online App recommendations, we build an *App hash tree* to efficiently look up Apps. Finally, we evaluate our mobile App recommendation approach with extensive experiments on a large-scale real-world data set collected from Google Play, which contains 170,753 mobile Apps. The experimental results clearly validate the effectiveness and efficiency of our approach in terms of different evaluation metrics.

2. PROBLEM FORMULATION

In this section, we first introduce some preliminaries about the security/privacy problems of mobile Apps, and then introduce the framework of the proposed mobile App recommender system with security and privacy awareness.

Table 1: Examples of data access permissions.

Type	Permission ID	Description
String	`ACCESS_FINE_LOCATION`	Allows an application to access fine (e.g., GPS) location.
String	`READ_CONTACTS`	Allows an application to read the user's contacts data.
String	`READ_SMS`	Allows an application to read the user's SMS messages.
String	`READ_CALENDAR`	Allows an application to read the user's calendar data.
String	`READ_CALL_LOG`	Allows an application to read the user's call log.

2.1 Preliminaries

The most advanced mobile operating systems, such as Apple IOS, Google Android, and Microsoft Windows Phone, implement a sandbox which provides the security and privacy policy for the third-party mobile Apps. To be specific, these operating systems isolate Apps from each other and the resources, thus feature a permission system [7]. To access the personal data in users' mobile devices, the permission system will convey users to grant corresponding data access permissions explicitly (e.g., IOS) or implicitly (e.g., Android) for each mobile App. Actually, these data access permissions may enter some sensitive resources in mobile users' personal data, such as their locations or contact lists. For instance, Table 1 illustrates some examples of data access permissions in the Android system [1]. We can see that all these listed permissions contain potential security risks. For example, an App, which requests `READ_CALENDAR` and `READ_SMS` permissions, may access users' personal calendar and short messages. This may not be comfortable for a business man due to the risks of leaking confidential information.

Indeed, all these data access permissions can be categorized into different levels with respect to their potential security risks. For example, as defined by Android Developers [1], there are three different threat levels for managing data access permissions,

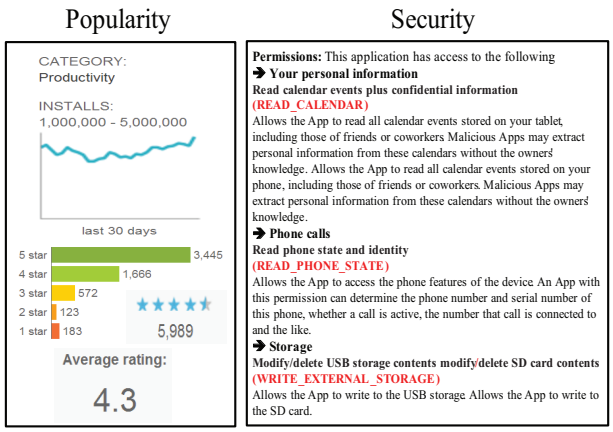


Figure 2: A motivating example.

- **Normal permissions** give an App access to isolated App level features, with the minimal risk to other applications, the system, or the user access (e.g., the permission to set screen wallpaper).
- **Dangerous permissions** give an App access to private user data or control over the device, with a potential risk that can negatively impact the user (e.g., the permission to have the user's current location).
- **Signature/System permissions** give an App access to the dangerous privileges, which need system signature certifications such as the ability to control the system process (e.g., the permission to delete Apps).

To provide better services to users and gain more downloads of Apps, mobile App developers try to request more and more data access permissions, which can help to implement the intelligent applications, such as social sharing services. However, these services may result in potential security and privacy risks. For example, Figure 2 shows an example of a mobile App in the Android market, which contains both popularity and security information. In this figure, we can observe that this App may request the permission of reading the users' calender (i.e., `READ_CALENDAR`), reading phone states (i.e., `READ_PHONE_STATE`) and external USB/SD card storage (i.e., `WRITE_EXTERNAL_STORAGE`). Although this is a quite popular App according to user ratings and the download information, it may still contain the potential risk of leaking user information. For instance, if this App is controlled by a Trojan, it could gather users' calender information and phone numbers, then upload the information into external USB disk or SD card (when connected) via the above permissions. However, to the best of our knowledge, this kind of security risks is not taken into account in most existing mobile App recommender systems. Indeed, they only focus on the Apps' popularity information (e.g., user ratings). Thus, we aim on developing a mobile App recommender system with security and privacy awareness.

2.2 The Recommendation Framework

Here, we first formally define the problem of mobile App recommendations with security and privacy awareness, and then show the recommendation framework.

Definition 1 (Problem Statement). *Given a category label c, and a set of Apps $A = \{a\}$, each of which contains a set of data access permissions $\{p_i\}$, profile information (e.g., category, popularity), the goal of mobile App recommendation with security and privacy awareness is to build an optimal ranked list of Apps in category c based on both the Apps' popularity and users' security preferences.*

Indeed, the above problem statement raises two issues:

- How to mine the security risks of Apps and produce a ranked list $\Lambda^{(Risk)} = \{a | a \in c\}$ according to their **risk scores** $Risk(a)$, where a is ranked higher than a^* if and only if $Risk(a) > Risk(a^*)$.
- How to combine the risk based ranked list $\Lambda^{(Risk)}$ with the popularity based ranked list $\Lambda^{(Pop)}$ to produce final ranking so as to meet various expectations of users, who have different security and privacy concerns.

While it is appealing to provide mobile App recommendations with security and privacy awareness, it is a non-trivial task to effectively discover and evaluate the security risks of Apps, and produce desirable ranking of Apps by considering both Apps' popularity and users' security preferences. In addition, there are often many different security preferences of mobile users, and a huge number of Apps as candidates for recommendations. Thus, how to efficiently manage Apps for recommendation is also an open question. To that end, in this paper, we propose a novel recommendation framework to solve these problems.

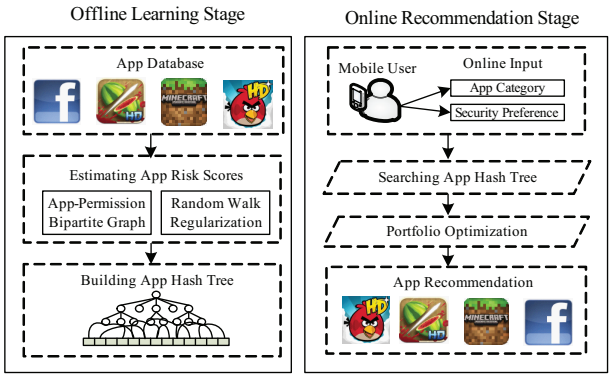


Figure 3: The recommendation framework.

Figure 3 shows the proposed recommendation framework, which consists of two stages. The *offline learning stage* automatically learns the risk scores for Apps by leveraging the random walk regularization with an *App-permission bipartite graph*, and forms an App hash tree from the App data set for efficiently managing Apps. The *online recommendation stage* matches the given mobile users' security preferences and App categories according to the App hash tree, ranks the candidate Apps with respect to both Apps' popularity and users' security preferences by leveraging the modern portfolio theory for recommendations.

3. ESTIMATING RISK SCORES FOR MOBILE APPS

Generally speaking, the risk score reflects the security level of an App. The smaller the score is, the more safe the App is. According to the above discussion, we can know the security risks are essentially caused by the data access permissions of Apps. Thus, an intuitive approach for measuring the risks of Apps is to directly check each of the dangerous permissions they request. However, there are many critical challenges along this line, which make the problem

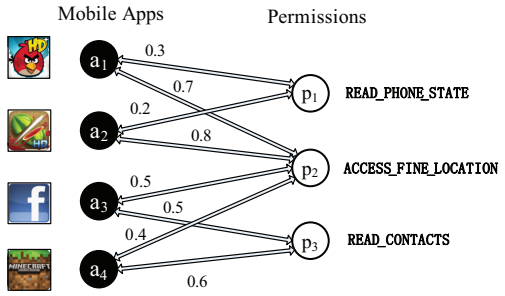


Figure 4: An example of the bipartite graph.

still under-addressed. First, it is hard to explicitly define a risk function with respect to different permissions for evaluating the potential risks of mobile Apps, since the permissions are often very ambiguous and poorly understood [5, 8]. For example, we observe that although some permissions are dangerous (e.g., location related permissions), they are commonly used in the Apps of some categories (e.g., navigation Apps). Second, the latent relationships between Apps and permissions should be taken into consideration, since similar Apps (permissions) should have similar risk scores. Finally, we should develop a scalable approach to refine risk scores, since rich external knowledge can be leveraged for evaluating potential risks of Apps. For example, some external risk reports, the state-of-the-art security models in relevant domains as well as the prior knowledge from domain experts can be leveraged for improving the performance of ranking App risks. To deal with the above challenges, in this paper, we propose a regularization approach based on a bipartite graph, which can learn the security risk of mobile Apps automatically without relying on any predefined risk function. Particularly, we develop an *App-permission bipartite graph* to build the connections between Apps and permissions, which is defined as follows.

Definition 2 (App-permission Bipartite Graph). *The graph can be denoted as $G = \{V, E, W\}$. $V = \{V^a, V^p\}$ is the node set, where $V^a = \{a_1, \cdots, a_M\}$ denotes the set of Apps and $V^p = \{p_1, \cdots, p_N\}$ denotes the set of permissions. E is the edge set, where $e_{ij} \in E$ exists if and only if a_i requests the permission p_j. W is the edge weight set, where each $w_{ij} \in W$ represents the weight of e_{ij} and denotes the probability that a_i will request p_j.*

Figure 4 shows an example of App-permission bipartite graph. Intuitively, the weight w_{ij} can be estimated by the permission records of all Apps in a_i's category. Specifically, we can compute the weight by

$$w_{ij} = \frac{f_{ij}}{\sum_{e_{ik} \in E} f_{ik}}, \tag{1}$$

where f_{ij} is the number of Apps in category c ($a_i \in c$) requesting permission p_j. Furthermore, we can denote each App a_j and permission p_j as vectors $\overrightarrow{a_i} = \{w_{i1}, \cdots, w_{iN}\}$ and $\overrightarrow{p_j} = \{w_{1j}, \cdots, w_{Mj}\}$, respectively. Accordingly, we define the latent similarity between Apps a_i and a_j by the Cosine distance,

$$s^a_{ij} = Cos(\overrightarrow{a_i}, \overrightarrow{a_j}) = \frac{\overrightarrow{a_i} \cdot \overrightarrow{a_j}}{\|\overrightarrow{a_i}\| \cdot \|\overrightarrow{a_j}\|}. \tag{2}$$

Similarly, we define the latent similarity between permissions p_i and p_j as $s^p_{ij} = Cos(\overrightarrow{p_i}, \overrightarrow{p_j})$.

To estimate App risk scores with the App-permission bipartite graph, we first define two scores $Risk(a_i)$ and $Risk(p_j)$ for node $a_i \in V^a$ and $p_j \in V^p$, respectively. Intuitively, $Risk(a_i)$ is the objective App risk score and $Risk(p)$ is the global permission risk score. Second, we develop a regularization framework by regularizing the smoothness of the above two scores over the bipartite graph. Specifically, if we denote $Risk(a_i)$ as R^a_i and $Risk(p_j)$ as R^p_j, we define a cost function as follows,

$$\begin{aligned}\mathcal{Q}(a,p) = \frac{\lambda}{2} \cdot \Big\{ \sum_i \Big\| R^a_i - \widetilde{R}^a_i \Big\|^2 + \sum_j \Big\| R^p_j - \widetilde{R}^p_j \Big\|^2 \Big\} + \\ \frac{\mu}{2} \cdot \Big\{ \sum_{i,j} s^a_{ij} \Big\| R^a_i - R^a_j \Big\|^2 + \sum_{i,j} s^p_{ij} \Big\| R^p_i - R^p_j \Big\|^2 \Big\} + \\ \frac{1}{2} \cdot \sum_{i,j} w_{ij} \Big\| R^a_i - R^p_j \Big\|^2,\end{aligned} \tag{3}$$

where λ and μ are the regularization parameters, $\widetilde{R}^a_i$ and $\widetilde{R}^p_j$ are the *prior risk scores* derived from *external knowledge*.

Intuitively, this cost function is formed by three parts. The first part controlled by λ defines the constraint that the two risk scores should fit prior knowledge. The second part controlled by μ defines the global consistency of the refined risk scores over the graph. Specifically, it satisfies that, if two Apps (permissions) have high latent similarity, their risk scores should be similar. The third part is the smoothness constraint between Apps and permissions, which guarantees that, if an App has high probability to request a specific permission, their risk scores should be similar. Therefore, the problem of estimating risk scores is converted to the optimization problem of finding optimal R^a_i and R^p_j to minimize the cost function $\mathcal{Q}$. In this paper, we exploit the classic gradient descent method to solve this problem. Specifically, we first assign values to $R^a_i = 1/M$ and $R^p_j = 1/N$ and iteratively update them by setting the following differentiated results to zero.

$$\begin{aligned}\frac{\partial \mathcal{Q}}{\partial a_i} &= \lambda(R^a_i - \widetilde{R}^a_i) + \mu \sum_j s^a_{ij}(R^a_i - R^a_j) + \sum_j w_{ij}(R^a_i - R^p_j), \\ R^a_i &= \frac{\lambda \widetilde{R}^a_i + \mu \sum_j s^a_{ij} R^a_j + \sum_j w_{ij} R^p_j}{\lambda + \mu \sum_j s^a_{ij} + \sum_j w_{ij}}.\end{aligned} \tag{4}$$

$$\begin{aligned}\frac{\partial \mathcal{Q}}{\partial p_j} &= \lambda(R^p_j - \widetilde{R}^p_j) + \mu \sum_i s^p_{ij}(R^p_j - R^p_i) + \sum_i w_{ij}(R^p_j - R^a_i), \\ R^p_j &= \frac{\lambda \widetilde{R}^p_j + \mu \sum_i s^p_{ij} R^p_i + \sum_i w_{ij} R^a_i}{\lambda + \mu \sum_i s^p_{ij} + \sum_i w_{ij}}.\end{aligned} \tag{5}$$

After each iteration, all the values of R^a_i and R^a_j will be normalized again, i.e., $\|R^a\|_1 = 1$ and $\|R^p\|_1 = 1$. Finally, we can obtain the optimal risk scores after the results converge.

How to assign prior risk scores $\widetilde{R}^a_i$ and $\widetilde{R}^p_j$ from external knowledge is an open question. In practice, some intuitive solutions include inviting domain experts for assigning risk scores, building a security classifier through external risk reports, or exploiting state-of-the-art security models in relevant domains. In this paper, as an attempt, we leverage the probabilistic approach PNB (Naive Bayes with information Priors) proposed in [14] for this task, which is based on the scoring scheme, and thus can be directly adopted by our regularization framework. Specifically, PNB aims to learn a Naive Bayes model with parameter θ that can best explain the generative process of permissions, i.e., $P(p_j|\theta)$. In this model, the parameter θ is assumed to follow the Beta prior $Beta(\theta; \alpha_0, \beta_0)$, and the probability can be estimated by

$$P(p_j|\theta) = \frac{\sum_i^M x_{i,j} + \alpha_0}{M + \alpha_0 + \beta_o}, \tag{6}$$

where M is the total number of Apps and $x_{i,j}$ is a binary function which is equal to 1 (i.e., a_i requests the permission p_j) or 0 (i.e., a_i does not request the permission p_j). Particularly, PNB also defines three categories of permissions with respect to their threat levels (i.e., similar as the preliminaries in Section 2), and each category has a specific $Beta(\theta; \alpha_0, \beta_0)$ as informative priors. Therfore, the risk scores of permission p_j and App a_i can be estimated by $\widetilde{R}_j^p = -\ln P(p_j|\theta)$ and $\widetilde{R}_i^a = -\ln P(p_1, \cdots, p_k|\theta)$, where each $p_k \in a_i$. Note that, both $\widetilde{R}_i^a$ and $\widetilde{R}_j^p$ are normalized before learning our regularization framework. Although PNB is a straightforward approach that cannot solve all the challenges mentioned before, its effectiveness on ranking risks of Apps has been well proved. Therefore, using PNB as prior knowledge in our regularization framework is appropriate.

4. RANKING FOR MOBILE APP RECOMMENDATION

Algorithm 1 Automatic Detection of Security Levels

Input: The set of Apps $A = \{a_i\}$; Parameter δ;
Output: The set of security levels Ψ;

```
1: Rank A in descending order according to Risk(a);
2: L = ∅;
3: for each i ∈ [1, |A|] do
4:    A* = L ∪ {A[i]};
5:    calculate CV(A*) in terms of Risk(a) (a ∈ A*);
6:    if (CV(A*) > δ) then
7:       Ψ ∪ = L; L = ∅ is a new level;
8:    else
9:       L∪ = {A[i]};
10:   end if
11: end for
12: return Ψ
```

After computing the risk score for each mobile App, we can rank Apps in ascending order with respect to their risk scores for recommendations. Moreover, if some Apps have the same risk scores, they will be further ranked according to popularity scores (e.g., overall rating). However, for real-world App recommendation services, users may have difficulties to get clear perception about the risks of ranked Apps. A promising way to help users understand the different risks of Apps is to categorize the risks into discrete levels (e.g., Low, Medium, High). In fact, people often describe their perception about risk or security with such discrete levels. Therefore, in this paper, we further group Apps into different clusters, each of which has the same security level (e.g, Low or High). However, it is not easy to get an accurate and appropriate segmentation of Apps with respect to their risk scores due to the lack of appropriate benchmarks.

To solve the above problem, we develop a Coefficient of Variation (CV) based approach to automatically segment mobile Apps. The main idea of this approach is that two adjacent Apps in the globally ranked list are assigned with different security levels, if their risk scores have dramatic differences, which can be captured by the CV, i.e., $\frac{variance}{mean}$, of their risk scores. The detailed segmentation algorithm is shown in Algorithm 1. The parameter δ is a threshold used for determining the dramatic difference of CV. After segmentation, the Apps at lower security levels have higher security risk.

Now, we are able to recommend Apps for users. Specifically, given a specific security level L^* and a category c, we can treat all the Apps in category c with security $L \geq L^*$ as candidates. Intuitively, there are two types of ranking principles for recommending Apps.

- ***Security Principle***: We first rank App candidates in ascending order by their risk scores, and Apps have the same scores will be further ranked by popularity scores (e.g., overall rating).
- ***Popularity Principle***: We first rank App candidates in descending order by their popularity scores (e.g., overall rating), and Apps have the same popularity scores will be further ranked by risk scores.

Furthermore, we need to strike a balance between users' security preferences and Apps' popularity for recommendations. To achieve such a balance, we also propose a ***hybrid principle*** for App recommendations, which is based on the modern portfolio theory [16]. The portfolio theory is originally proposed in the field of finance, which focuses on the investment problem of financial market. For example, an investor often wants to select a portfolio of n stocks with a fixed investment budget, which will provide the maximum future return and the minimum risk. In our problem, the stocks can be regarded as Apps, the future return and risk can be regarded as popularity and security risk of Apps.

Specifically, an App portfolio Υ can be represented by a collection of n Apps with a corresponding weight w_i assigned to each App a, i.e.,

$$\Upsilon = \big\{(a_i, w_i)\big\}, \quad s.t. \ \sum_i w_i = 1. \tag{7}$$

Indeed, the weight w_i in finance is the percentage of the budget invested in the i-th stock. According to the discussion in [19], the weight w_i in our problem indicates how much attention the recommender system wants the target user to pay on the App a_i. Therefore, the weights can be used to determine the ranks of Apps; that is, Apps should be ranked by the descending order of their weights. Before obtaining the weights, we first define the future return of the App portfolio as $\mathbb{E}[\Upsilon]$, which can be computed by

$$\mathbb{E}[\Upsilon] = \sum_i^n w_i \cdot \Delta_i^{-1}, \tag{8}$$

where Δ_i is the rank of App a_i in the popularity based ranked list $\Lambda^{(Pop)}$. Also, we define the future risk of the App portfolio as $\mathbb{R}[\Upsilon]$, which can be computed by the following function [12],

$$\mathbb{R}[\Upsilon] = \sum_i^n (w_i^2 \nabla_i^{-2} + 2 \sum_{j=i+1}^n w_i w_j \nabla_i^{-1} \nabla_j^{-1} J_{ij}), \tag{9}$$

where ∇_i is the rank of App a_i in the risk based ranked list $\Lambda^{(Risk)}$, and J_{ij} is the risk correlation between Apps a_i and a_j. Here, we estimate J_{ij} according to the similarity of requested permissions. For any two Apps, the more common permissions are requested, the higher risk similarity they have. To this end, we compute J_{ij} using Jaccard coefficient between Apps a_i and a_j by,

$$J_{ij} = \frac{N_{ij}}{N_i + N_j - N_{ij}}, \tag{10}$$

where N_i is the number of permissions requested by App a_i, and N_{ij} is the number of common permissions requested by two Apps a_i and a_j.

In our problem, the objective is to learn a set of App weights $\mathbf{w}$ for maximizing the future return and minimizing

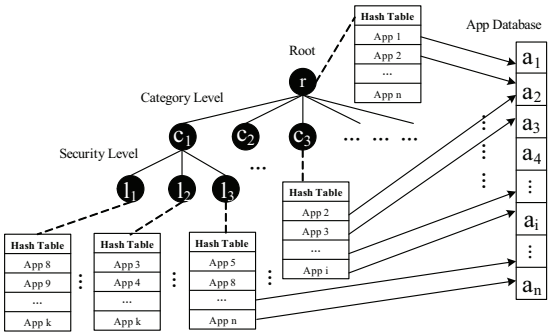


Figure 5: An example of the App hash tree.

the risk of the App portfolio Υ that consists of recommendation candidates (App candidates), i.e.,

$$\arg\max_{\mathbf{w}} \quad \mathbb{E}[\Upsilon] - b \cdot \mathbb{R}[\Upsilon], \tag{11}$$

where b is a specified risk preference parameter, which is defined as the given security level L^* in our experiments. The above optimization problem can be solved by the efficient frontier based approach introduced in [19]. Specifically, we can obtain the optimal weight $\mathbf{w}^*$ by

$$\mathbf{w}^* = \frac{\begin{vmatrix} 1 & \mathbf{1}^T\boldsymbol{\Sigma}^{-1}\mathbf{E} \\ E^* & \mathbf{E}^T\boldsymbol{\Sigma}^{-1}\mathbf{E} \end{vmatrix} \boldsymbol{\Sigma}^{-1}\mathbf{1} + \begin{vmatrix} \mathbf{1}^T\boldsymbol{\Sigma}^{-1}\mathbf{1} & 1 \\ \mathbf{E}^T\boldsymbol{\Sigma}^{-1}\mathbf{1} & E^* \end{vmatrix} \boldsymbol{\Sigma}^{-1}\mathbf{E}}{\begin{vmatrix} \mathbf{1}^T\boldsymbol{\Sigma}^{-1}\mathbf{1} & \mathbf{1}^T\boldsymbol{\Sigma}^{-1}\mathbf{E} \\ \mathbf{E}^T\boldsymbol{\Sigma}^{-1}\mathbf{1} & \mathbf{E}^T\boldsymbol{\Sigma}^{-1}\mathbf{E} \end{vmatrix}}, \tag{12}$$

where $\boldsymbol{\Sigma}_{ij} = \nabla_i^{-1}\nabla_j^{-1}J_{ij}$, $\mathbf{E} = (\Delta_1^{-1}, \cdots, \Delta_n^{-1})^T$, and E^* can be computed by

$$E^* = \frac{(xz - y^2)^2 - 2b(x\mathbf{E} - y\mathbf{1})^T\boldsymbol{\Sigma}^{-1}(z\mathbf{1} - y\mathbf{E})}{2b(x\mathbf{E} - y\mathbf{1})^T\boldsymbol{\Sigma}^{-1}(x\mathbf{E} - y\mathbf{1})}, \tag{13}$$

where $x = \mathbf{1}^T\boldsymbol{\Sigma}^{-1}\mathbf{1}$, $y = \mathbf{1}^T\boldsymbol{\Sigma}^{-1}\mathbf{E}$, and $z = \mathbf{E}^T\boldsymbol{\Sigma}^{-1}\mathbf{E}$.

After ranking Apps with respect to three different principles, the final challenge is how to organize and index such a large number of Apps with respect to their security levels and categories. Indeed, in an online App recommender system, it is necessary to quickly response users' requests and efficiently manage Apps in its back-end servers. To this end, we propose a data structure for App retrieval, namely *App hash tree*. Figure 5 illustrates an example of an App hash tree, which contains two hierarchies, namely a category level and a security level. For each node in the tree, it holds a hash table to store the index of corresponding Apps. For example, the node "$Root \rightarrow c_1 \rightarrow l_3$" may store the index of all Apps belong to category c_1 and security level l_3. Note that the App hash tree can be easily built with some basic tree search algorithms (e.g., *Breadth-First-Search* in our experiments). Actually, the ranking results of Apps in each node can be computed offline and pre-stored in the corresponding nodes of the App hash tree. Therefore, during the online recommendation, the system can quickly look up the ranked list for recommendations to users based on their specific security levels and App categories.

5. EXPERIMENTAL RESULTS

In this section, we empirically evaluate the **S**ecurity and **P**rivacy aware mobile **App** **R**ecommendation (SPAR) approach with a large-scale real-world data set.

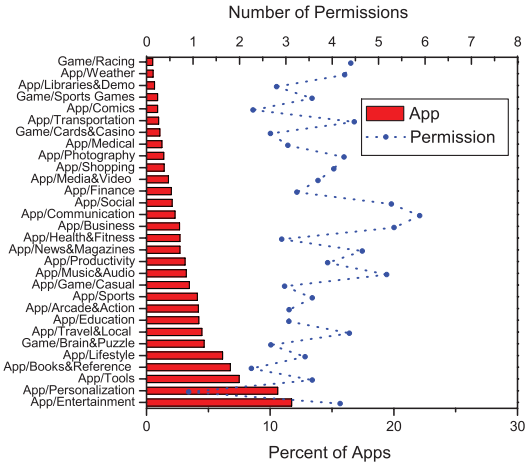


Figure 6: The percent of Apps and the average number of requested permissions by each App in different categories.

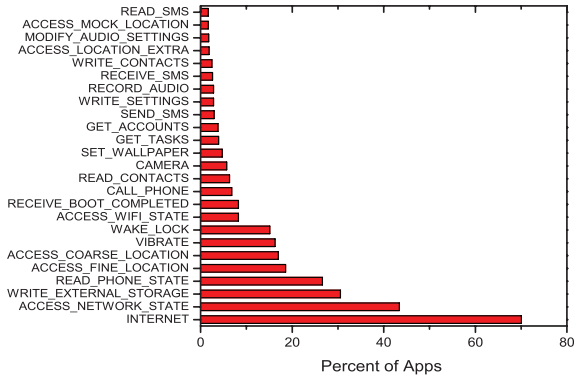
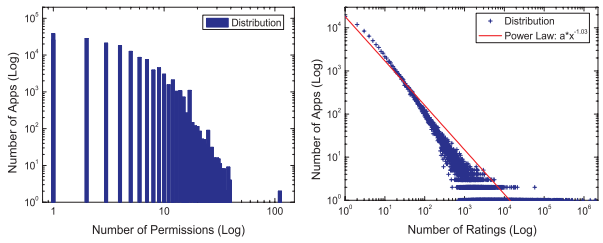


Figure 7: The top 25 most used permissions in our data set and the percent of Apps that request those permissions.

5.1 Experimental Data

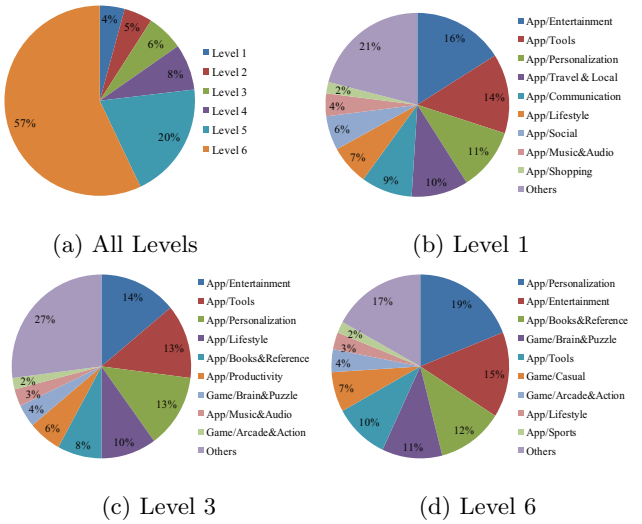
The experimental data were collected from Google Play (Android Market) [4] in 2012. This real-world data set includes 170,753 Apps in 30 App categories, and the Apps have 173 unique data access permissions. Particularly, the data set includes more than 25% Apps available at the Android Market, which totally includes 675,000 Apps as of the end of September 2012 [3].

Figure 6 and Figure 7 illustrate some statistics of the data set. Specifically, Figure 6 shows the percent of Apps and the average number of requested permissions by each App in different categories. In this figure, we can observe that Apps in categories "Communication", "Business" and "Social" request more permissions. Figure 7 shows the top 25 most requested permissions and the percent of Apps that request those permissions. In this figure, we can find that most of the Apps request the network and location related permissions. To further study the relationship between permissions and Apps, we show the distributions of the number of Apps with respect to the number of requested permissions in Figure 8 (a). We can see that most of the Apps only request few permissions, which may indicate that not many Apps have security risks. Figure 8 (b) shows the distribution of the number of Apps with respect to the number of their ratings. We can find that the distribution roughly follows the



(a) (b)

Figure 8: The distribution of the number of Apps w.r.t (a) the number of requested access permissions, and (b) the number of their ratings.



(a) All Levels (b) Level 1

(c) Level 3 (d) Level 6

Figure 9: The percent of (a) Apps and (b)-(d) App categories at different security levels.

power law. This indicates that only using App popularity for recommendation is not enough.

5.2 Evaluation of App Risk Scores

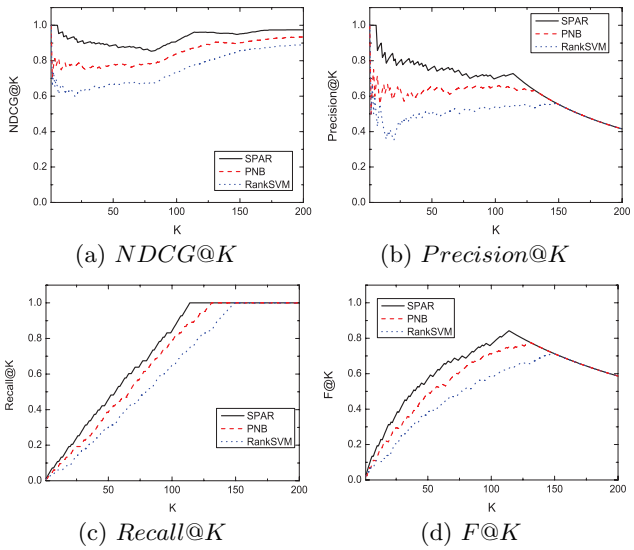
In this subsection, we evaluate the performances of estimating App risk scores and segmenting security levels.

5.2.1 App Security Levels

Specifically, we set the regularization parameters in Equation 3 as $\lambda = 0.5$ and $\mu = 1$, and the settings of PNB are similar as [14]. To segment Apps with respect to their risk scores, we empirically set $\delta = 0.01 \times CV(A)$ in Algorithm 1, where $CV(A)$ is the CV of all App risk scores.

Figure 9 (a) shows the percent of Apps with respect to 6 segmented security levels. We can see that level 6 (i.e., most secure) contains most Apps and the App numbers from level 1 to level 4 are relatively even, which indicate most Apps are secure while only a few Apps have security risks.

Figure 9 (b)-(d) show the percent of App categories at security levels 1, 3, and 6, respectively. In these figures, we can find that Apps with more permissions (e.g., Apps in categories "Tools", "Travel&Local", and "Communication") are more likely to have potential risks, and vice versa (e.g., Apps in categories "Personalization" and "Books&Reference"). Note that, since categories "Entertainment" and "Personalization" contain the largest portion of Apps in our data set, they always have high percent at all security levels.



(a) *NDCG@K* (b) *Precision@K*

(c) *Recall@K* (d) *F@K*

Figure 10: The performance of each approach w.r.t different metrics based on user judgment.

5.2.2 Evaluation of Ranking App Risk

Evaluation Baselines. We adopt two state-of-the-art baselines to evaluate the performances of our SPAR approach in terms of ranking App risks. To the best of our knowledge, there is only one relevant recent study [14], which can be directly leveraged for ranking App risks. Therefore, we leverage the recommended approach in this work as the first baseline. **Naive Bayes with information Priors** (PNB) [14] aims to learn a Naive Bayes model with parameter θ that can best explain the generative process of permissions, i.e., $P(p_i|\theta)$. Therefore, the risk scores of App a_i can be estimated by $\widetilde{R}_i^a = -\ln P(p_1, \cdots, p_k|\theta)$, where each $p_k \in a_i$. Particularly, this baseline is also used for estimating the prior risk scores in our regularization framework. Moreover, we also use a popular learning-to-rank approach as the second baseline for ranking App risks. **RankSVM** [9] aims to rank App risk by the RankSVM model. Specifically, we manually labeled 200 secure Apps and 200 insecure Apps according to some previous studies [6, 14, 20] as training data. For each App, we used its category, developer, and permissions as features to learn the ranking model.

Evaluation Metrics. Specifically, we set up the evaluation as follows. First, we implemented our SPAR approach and other baselines on all the Apps in the data set. For each approach, we selected 100 top ranked mobile Apps (i.e., most insecure), and 100 bottom ranked mobile Apps (i.e., most secure) in the result. Then, we merged all the selected Apps into a pool which includes 496 unique mobile Apps in our data set. For each App, we invited three users who are familiar with Android Apps to manually label these Apps with score 2 (i.e., **Insecure**), 1 (i.e., **Not Sure**), and 0 (i.e., **Secure**). Each user gave a proper label by comprehensively considering their own experiences (i.e., they can download and try all these Apps), the App profile and the comments from other users. After user evaluation, each App a is assigned a judgement score $f(a) \in [0, 6]$. Moreover, we computed the Cohen's kappa coefficient [2] between each pair of evaluators to estimate the inter-evaluator agreement. The values of Cohen's kappa coefficient are between 0.67 to 0.72 in the user evaluation, which indicate the substantial agree-

ment [11]. Finally, we further ranked the 496 Apps by each approach, and obtained three ranked lists of Apps. Thus, we can exploit the popular metric Normalized Discounted Cumulative Gain (NDCG) for determining the ranking performance of each approach. Specifically, the discounted cumulative gain given a cut-off rank K can be calculated by

$$DCG@K = \sum_{i=1}^{K} \frac{2^{Rel(a_i)} - 1}{\log_2(1+i)},$$

where $Rel(a_i) = f(a_i)$ is the relevance score. The $NDCG@K$ is the $DCG@K$ normalized by the $IDCG@K$, which is the $DCG@K$ value of the ideal ranking list of the returned results. In other words, we have

$$NDCG@K = \frac{DCG@K}{IDCG@K}.$$

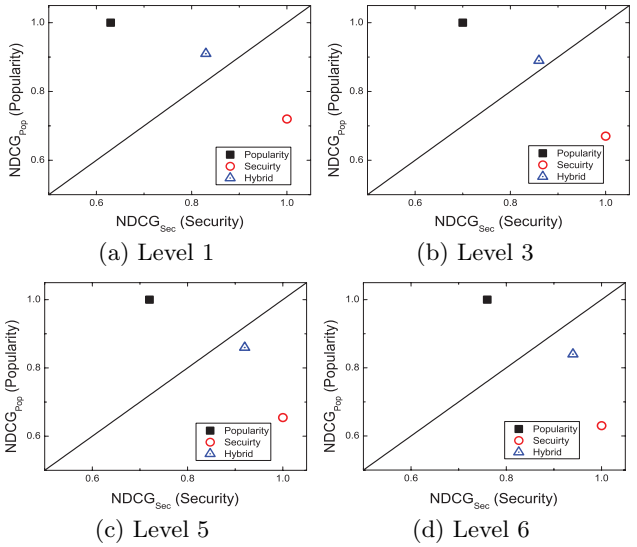
$NDCG@K$ indicates how well the ranked order of the given Apps returned by an approach with a cut-off rank K. A larger $NDCG@K$ value indicates the better ranking performance. Particularly, if we treat the 83 commonly agreed insecure Apps (i.e., $f(a) = 6$) as the ground truth, we can evaluate each approach with the widely-used metrics, namely $Precision@K$, $Recall@K$, and $F@K$.

Overall Performances. Figure 10 shows the results of each approach with respect to four different evaluation metrics. In this figure, we can see that SPAR consistently outperforms other baselines and the improvement is more significant for smaller K. These results clearly validate the effectiveness of our regularization based approach. Particularly, the performances of PNB can be refined during the regularization on the bipartite graph. Also, SPAR and PNB outperform RankSVM, which indicates the straightforward learning-to-rank approach is not enough for estimating App risks. Indeed, the performances of learning-to-rank approaches mainly rely on the effectiveness of feature extraction. Based on the above observations, we can argue that SPAR is an appropriate approach for estimating App risks.

Case Study. This evaluation benchmark is based on some prior knowledge from other previous studies. As reported by Zhou *et al* [20], there are 13 Apps which may leak private information according to the TaintDroid system [6]. Here, we select 6 of them (i.e., *Horoscope, Layar, Trapster, Wertago, Astrid Task and DasTelefonbuch*), which are included in our data set, to evaluate SPAR and other baselines. Indeed, we study whether each approach can find these insecure Apps with high risk ranks, since a good approach should have the capability of capturing these suspicious Apps. Table 2 shows the top percentage position of each App in the ranked list returned by each approach. We can see that SPAR can rank those insecure Apps into higher positions than other baselines. Specially, all of these six Apps are categorized into low security levels (i.e., L_1 and L_2) by the segmentation approach, which also validates the effectiveness of our approach.

Table 2: The reported insecure mobile Apps.

	SPAR	PNB	RankSVM
Horoscope	**2.64%**	5.41%	7.13%
Layar	**5.34%**	7.21%	11.81%
Trapster	**6.21%**	9.34%	12.33%
Wertago	**2.72%**	4.89%	8.37%
Astrid Task	**8.09%**	11.29%	13.32%
DasTelefonbuch	**6.18%**	11.71%	14.38%



(a) Level 1 (b) Level 3

(c) Level 5 (d) Level 6

Figure 11: The recommendation performances of different ranking principles.

5.3 Evaluation of App Recommendation

Here, we evaluate the recommendation performances of our approach SPAR. Particularly, we use the average rating as the popularity score for each App and the parameter b in Equation 11 equals to the given security level in experiments.

5.3.1 Recommendation Performances

Since our App recommender system is non-personalized, there is no personal data could be used for evaluation. Also, there is no ground truth for us to evaluate which recommendation results really meet users' information needs. Thus, in this paper, we focus on evaluating our recommendation approach SPAR by checking whether it can strike a balance between App popularity and user's security preferences.

Specifically, there are three different ranking principles in our recommendation approach, i.e., popularity, security and hybrid principles. Given an App category and security level, each principle can generate a ranked App list as the recommendation result. Here, we propose to use two metrics $NDCG_{Pop}$ and $NDCG_{Sec}$ to evaluate the the performance of each recommendation result. Compared with traditional $NDCG$, the relevance scores of $NDCG_{Pop}$ and $NDCG_{Sec}$ are set to the popularity score and the reciprocal of risk score, respectively. Intuitively, if a recommendation result has higher $NDCG_{Pop}$ ($NDCG_{Sec}$), it has more emphasis on App popularity (App Security). Figure 11 shows the average recommendation performance across all App categories with respect to different ranking principles and security levels. From the results, we can observe that the hybrid principle can rank Apps with a trade-off between popularity and security, which means the recommended Apps are both popular and secure. Also, with the increase of security levels, the recommendation results have more emphasis on App security than popularity.

5.3.2 A Case Study

To further evaluate the recommendation performances of different ranking methods, we study five Apps in category "*App/Lifestyle*", which are "Weterago", "BeNaughty", "Moment Diary", "SimplyNoise" and "Bedside". Particularly,

Table 3: The case study of App recommendation.

	Recommendation
SEC	SimplyNoise,Moment Diary,Bedside,BeNaughty,Weterago
POP	Weterago,Bedside,Moment Diary,BeNaughty,SimplyNoise
H-1	Bedside,Moment Diary,Weterago,BeNaughty,SimplyNoise
H-3	Moment Diary,Bedside,BeNaughty,SimplyNoise
H-5	Moment Diary,SimplyNoise,Beside
H-6	SimplyNoise,Moment Diary

"Weterago" is one of the reported insecure Apps, and "SimplyNoise" is an App without requesting data access permissions. Table 3 shows the recommendation results, where SEC recommends most secure Apps based on risk scores with security level 1; POP is based on popularity scores (i.e., average ratings) with the security level 1; H-1, H-3, H-5 and H-6 denote the hybrid principle based recommendation with security levels 1, 3, 5 and 6, respectively. From these results, we can observe that the popularity-based method recommends insecure App "Weterago" in the first position, while it has the highest risk score. In contrast, if only using risk scores to recommend Apps, some unpopular Apps (e.g., SimplyNoise) will be ranked higher. Furthermore, we can observe that H-3, H-5, H-6 do not recommend all five Apps for users. The reason is that these methods only take Apps with security levels higher than the given levels as candidates. Finally, we can see that the App "Moment Diary" is ranked the highest by H-3, H-5, since the hybrid principle can reach some balance between popularity and security for App recommendation.

5.4 Efficiency and Scalability

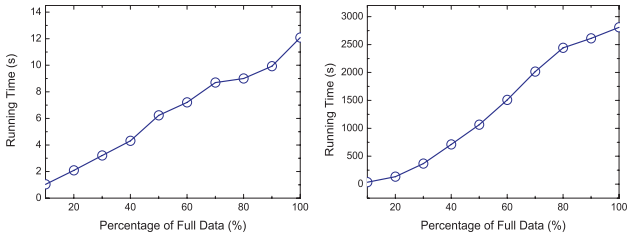
Our approach consists of an offline stage and an online stage. In the offline stage, the computational cost mainly comes from two parts: the computation of regularization for estimating risk scores, and the computation for security level segmentation and building the App hash tree. To evaluate the efficiency and scalability of our approach, we test the running time of each part on different segmentation of the entire data set (i.e., 10%,..., 100%) to illustrate the scalability of our approach. All the tests were conducted on a 3.4GHZ×8-Core CPU, 8G main memory PC. Figure 12 shows the running time of each part with respect to different input data size. We can see that the computation times are almost linear with the size of input data. Thus, our approach is scalable in the offline stage.

In the online stage, given a security level and App category, the recommender system will return the ranked list of Apps to user according to different recommendation principles. Indeed, since the popularity scores (e.g., overall rating) and risk scores can be obtained in the offline stage, and the portfolio optimization for hybrid principle has a close-form solution (e.g., Equation 12), the computational cost in online stage is relatively low. In particular, as discussed in Section 4, the main ranking process can be conducted in advance and pre-stored in the App hash tree. In this case, the online recommendation process will be very fast.

6. RELATED WORK

Generally speaking, the related works of this study can be grouped into two categories.

The first category is about mobile App security. Indeed, many previous studies about security and privacy issues of mobile Apps have been reported. For example, Enck *et al.* [6] proposed a malware detection system named TaintDroid, which can provide efficient real-time analysis of other third party mobile Apps through the monitor of their data



(a) (b)

Figure 12: The running time of (a) each iteration of regularization, and (b) security level segmentation and building the App hash tree.

access behavior. Luo *et al.* [13] discussed the problem of attacks on WebView in the Android system, analyzed the fundamental causes and proposed some potential solutions. To tame the information-stealing mobile Apps, Zhou *et al.* [20] proposed a new privacy model for Android system. Also, they developed a system named TISSA as security middleware to implement this model. Enck *et al.* [7] developed a rule-based certification model and system named Kirin, which can perform lightweight certification of mobile Apps at install time. Indeed, more and more advanced mobile Apps are committed to provide intelligent services for users by requesting various access permissions of users' personal data. To understand these data access permissions, Au *et al.* [5] surveyed the permission systems of several popular smart phone operating systems, such as Apple IOS, and Android. They also discussed the problem of permission over-declaration and proposed some insightful directions of relevant research. Similarly, Felt *et al.* [8] studied the permission requests of over 900 mobile Apps in Android system, and developed a tool named Stowaway to detect the over-privilege in compiled Android Apps.

However, these approaches are very hard to be implemented in practice, since it is not a trivial task to efficiently and accurately detect the malware codes for each mobile App and users often do not want some security software to frequently scan their devices. Recently, Peng *et al.* [14] proposed a novel approach with various probabilistic models for ranking Apps with respect to their risk scores. Although this approach is straightforward and not scalable for external knowledge, it is effective for estimating App risk. Therefore, we propose to leverage this approach for assigning prior risk scores in our regularization framework.

Another category is about mobile App recommendation, which aims to facilitate the choice of mobile users. For example, Yan *et al.* [17] developed a collaborative filtering based mobile App recommender system, namely Appjoy. Different from other mobile App recommender systems, the Appjoy is based on users' App usage records to build preference matrix but not explicit user ratings. However, sometimes the App usage records are very sparse. To solve this problem, Shi *et al.* [15] studied several recommendation models and proposed a content based collaborative filtering model named Eigenapp for recommending Apps in their Web site Getjar. Also, some researchers studied the problem of exploiting enriched contextual information for mobile App recommendation. For example, Yu *et al.* [18] proposed a novel personalized context-aware recommender system by analyzing mobile user's context logs. The proposed approach is based on Latent Dirichlet Allocation topic model and scalable for multiple contextual features. Furthermore, Zhu *et al.* [21] proposed a uniform framework for personalized context-aware

recommendation, which can integrate both context independency and dependency assumptions. The framework can mine user's personal context-aware preferences for mobile App recommendation from the context logs of many mobile users. However, all the above recommendation approaches do not take consideration of the potential security/privacy risk of mobile Apps, which motivates our novel mobile App recommender system with security and privacy awareness.

7. CONCLUDING REMARKS

In this paper, we developed a mobile App recommender system with security and privacy awareness. Specifically, without relying on any predefined risk functions, we designed a scalable and automatic approach for estimating the security risks of Mobile Apps. An unique perspective of this approach is the creative use of external knowledge as prior scores and the regularization techniques in an App-permission bipartite graph. Moreover, to consider both Apps' popularity and users' security preferences for recommendations, we introduced a flexible App recommendation method based on the modern portfolio theory. Particularly, we also developed an App hash tree to efficiently look up Apps in recommendation. Finally, the experiments on a large-scale real-world data set clearly validated the effectiveness and efficiency of the proposed recommendation framework.

8. ACKNOWLEDGEMENTS

This work was supported in part by grants from National Science Foundation for Distinguished Young Scholars of China (Grant No. 61325010), Natural Science Foundation of China (NSFC, Grant No. 71329201), and National High Technology Research and Development Program of China (Grant No. SS2014AA012303). This work was also partially supported by grants from National Science Foundation (NSF, Grant No. CCF-1018151 and IIS-1256016), and UNC Charlotte Faculty Research Grants 2014-2015.

9. REFERENCES

[1] http://developer.android.com/.

[2] http://en.wikipedia.org/wiki/cohen's_kappa.

[3] http://en.wikipedia.org/wiki/google_play.

[4] https://play.google.com/apps.

[5] K. W. Y. Au, Y. F. Zhou, Z. Huang, P. Gill, and D. Lie. Short paper: a look at smartphone permission models. In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, SPSM '11, pages 63–68, New York, NY, USA, 2011. ACM.

[6] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. In *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, OSDI'10, pages 1–6, Berkeley, CA, USA, 2010. USENIX Association.

[7] W. Enck, M. Ongtang, and P. McDaniel. On lightweight mobile phone application certification. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pages 235–245, New York, NY, USA, 2009. ACM.

[8] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner. Android permissions demystified. In *Proceedings of the 18th ACM conference on Computer and communications security*, CCS '11, pages 627–638, New York, NY, USA, 2011. ACM.

[9] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 133–142, New York, NY, USA, 2002. ACM.

[10] A. Karatzoglou, L. Baltrunas, K. Church, and M. Böhmer. Climbing the app wall: enabling mobile app discovery through context-aware recommendations. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, CIKM '12, pages 2527–2530, New York, NY, USA, 2012. ACM.

[11] E.-P. Lim, V.-A. Nguyen, N. Jindal, B. Liu, and H. W. Lauw. Detecting product review spammers using rating behaviors. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 939–948, New York, NY, USA, 2010. ACM.

[12] C. Luo, H. Xiong, W. Zhou, Y. Guo, and G. Deng. Enhancing investment decisions in p2p lending: An investor composition perspective. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 292–300, New York, NY, USA, 2011. ACM.

[13] T. Luo, H. Hao, W. Du, Y. Wang, and H. Yin. Attacks on webview in the android system. In *Proceedings of the 27th Annual Computer Security Applications Conference*, ACSAC '11, pages 343–352, New York, NY, USA, 2011. ACM.

[14] H. Peng, C. Gates, B. Sarma, N. Li, Y. Qi, R. Potharaju, C. Nita-Rotaru, and I. Molloy. Using probabilistic generative models for ranking risks of android apps. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 241–252, New York, NY, USA, 2012. ACM.

[15] K. Shi and K. Ali. Getjar mobile application recommendations with very sparse datasets. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, pages 204–212, New York, NY, USA, 2012. ACM.

[16] J. Wang and J. Zhu. Portfolio theory of information retrieval. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 115–122, New York, NY, USA, 2009. ACM.

[17] B. Yan and G. Chen. Appjoy: personalized mobile application discovery. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, MobiSys '11, pages 113–126, New York, NY, USA, 2011. ACM.

[18] K. Yu, B. Zhang, H. Zhu, H. Cao, and J. Tian. Towards personalized context-aware recommendation by mining context logs through topic models. In *Proceedings of the 16th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining - Volume Part I*, PAKDD'12, pages 431–443, Berlin, Heidelberg, 2012. Springer-Verlag.

[19] W. Zhang, J. Wang, B. Chen, and X. Zhao. To personalize or not: A risk management perspective. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 229–236, New York, NY, USA, 2013. ACM.

[20] Y. Zhou, X. Zhang, X. Jiang, and V. W. Freeh. Taming information-stealing smartphone applications (on android). In *Proceedings of the 4th international conference on Trust and trustworthy computing*, TRUST'11, pages 93–107, Berlin, Heidelberg, 2011. Springer-Verlag.

[21] H. Zhu, E. Chen, K. Yu, H. Cao, H. Xiong, and J. Tian. Mining personal context-aware preferences for mobile users. In *Proceedings of the IEEE 12th International Conference on Data Mining*, ICDM'12, pages 1212–1217, 2012.

[22] H. Zhu, H. Xiong, Y. Ge, and E. Chen. Ranking fraud detection for mobile apps: A holistic view. In *Proceedings of the 22Nd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '13, pages 619–628, New York, NY, USA, 2013. ACM.