Automatic Permission Optimization Framework for Privacy Enhancement of Mobile Applications

Yiting Qu, Suguo Du, Shaofeng Li, *Graduate Student Member, IEEE*, Yan Meng, *Graduate Student Member, IEEE*, Le Zhang, *Graduate Student Member, IEEE*, and Haojin Zhu, *Senior Member, IEEE*

***Abstract*—Mobile applications play a crucial role in the IoT system, which is experiencing unprecedented growth. However, users possessing little knowledge of permission configurations often accept app permission requests without reading them, which opens a backdoor for the potential adversaries to launch the future attacks. Proposing an automatic permission management scheme is an attractive solution to solve this issue, but since users have varying attitudes toward privacy, such a scheme would be neither straightforward nor user friendly. In this study, an automatic permission optimization framework, Permizer, is proposed to recommend different app permission configurations to users with different privacy preferences. Permizer estimates the permission risks and builds the permission-functionality mapping to each app, then regulates the relationship between permission and app functionality. Permizer is the first module to achieve a balance between privacy protection and app functionality under the personal privacy preference condition. Finally, we develop Permizer as a one-button service on the real-world Android OS with 58 apps. Case studies conducted on TikTok and Amazon Alexa also demonstrate its practicability and effectiveness.**

***Index Terms*—Mobile application, permission management, privacy protection, risk estimation.**

I. Introduction

THE Internet of Things (IoT) is a paradigm in which sensors, actuation devices, and computing devices are connected by a network such that each device in this network depends on and shares information with the others [1]. The deployment of IoT systems (e.g., smart homes, smart transportation, and smart healthcare) has risen dramatically; the GSMA [2] predicts that the global size of IoT connections will grow from 12 billion in 2019 to 25 billion in 2025. To facilitate the development of IoT systems, mobile applications (apps) must coordinate state-of-the-art devices (e.g., smart appliances and smart cars) manufactured by different vendors, provide user interfaces, and ensure the working logic of the system.

Despite the convenience brought by mobile applications, they also introduce serious privacy leakage issues [3]. In 2019, hundreds of thousands of users suffered breaches of private data via multiple companies, including Facebook, Orvibo, and Blur [4]. Among the various reasons of privacy leakage [5], [6], the most common reason is that app developers needlessly request permissions to gather user information, the so-called *over-privileged phenomenon* [7]. Such excessive permissions requesting is a pervasive phenomenon among mobile applications, which potentially undermines user privacy. One example is Baidu Map, a popular app used in smart transportation systems. It requests 13 sensitive permissions, including location, contact information and camera access, when starting the service. Although requesting location permission is reasonable for the app's navigation function, contact information, and camera will not be used in most scenarios. Granting these less relevant permissions is unnecessary and highly risky. To solve such a security issue, an efficient and user-friendly application permission management scheme is urgently needed.

Traditional permission management schemes were developed according to two principles: 1) *historical settings* and 2) *risk minimization*. The historical setting-based scheme responds to the application's new permission requests according to the user's previous permission management behavior [8], [9]. However, this scheme is fragile when the user has engaged in insecure historical behavior or is not security conscious (e.g., giving microphone permission to a spy app if the user always grants such permission to all existing apps). On the other hand, the risk minimization-based scheme alerts the user when a potentially sensitive permission (or permission set) is requested by any app [10], [11]. However, such frequent alerts reduce the user experience. To solve the limitation in the risk minimization-based scheme, many emerging schemes consider the app's functionalities when making permission assignment decisions [12]–[15]. For instance, these schemes will automatically grant location permission and disable message permission to a navigation app. However, for some specific permissions (e.g., microphone permission for navigation apps to enable voice interactive navigation), privacy-permissive and privacy-conservative users have completely different attitudes, and balancing the

Manuscript received August 29, 2020; revised October 9, 2020; accepted November 12, 2020. Date of publication November 25, 2020; date of current version April 23, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 71671114, Grant 61972453, and Grant 61672350. *(Corresponding author: Suguo Du.)*

Yiting Qu and Suguo Du are with the Antai College of Economics and Management, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: yiting001@sjtu.edu.cn; sgdu@sjtu.edu.cn).

Shaofeng Li, Yan Meng, Le Zhang, and Haojin Zhu are with the Shanghai Key Laboratory of Scalable Computing and Systems, Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: shaofengli@sjtu.edu.cn; yan_meng@sjtu.edu.cn; sj-16-zl@sjtu.edu.cn; zhu-hj@cs.sjtu.edu.cn).

Digital Object Identifier 10.1109/JIOT.2020.3039472

tradeoff between privacy protection and app functionality is still an unsolved research problem.

To solve the above drawbacks, we leverage the insight produced by combining the concept of user privacy preference (a willingness score when a user grants her permission) into permission management. More specifically, given a set of requested permissions from different users, we should recommend different, individual, and optimal permission configurations. However, to achieve an efficient permission management scheme, we need to address the following research challenges. First, given a specific permission, it is hard to quantify its risk among the permission set requested by apps. Second, for a mobile app, the inability to characterize the relationship between the permissions and the functionality of the app remains an open problem. Third, even if permission risk and the relationship between the permission and app function are obtained, for users with fluctuating privacy preferences, quickly adjusting the tradeoff between privacy protection and app functionalities remains a major challenge.

In this study, we propose an automatic permission management scheme, Permizer, to achieve optimal privacy decisions for mobile apps with different user privacy preferences. First, Permizer identifies which permissions are critical to privacy protection based on a statistical model built from two popular app permission data sets. Then, by applying natural language processing (NLP) techniques to the textual descriptions of the apps and permissions, Permizer successfully calculates the relatedness between the permission and the app's function. Finally, Permizer leverages the user's privacy preference as a weight parameter in our multiobjective optimization to personally configure an optimal recommended permission assignment. Permizer can serve as a one-button service that automatically grants permissions, and case studies on TikTok and Amazon Alexa demonstrate its feasibility and effectiveness.

The main contributions of this article can be summarized as follows.

1) Permizer is the first automatic permission management model that considers permission risk, app functionalities, and user privacy attitude at the same time.
2) We elaborate on how to estimate the permission risk, map the permission to the app function, and achieve a compromise between privacy protection and app functionality.
3) We implement Permizer on data sets containing 58 apps with three different user privacy preference levels. Furthermore, we select two apps (TikTok and Amazon Alexa) as case studies to demonstrate the effectiveness of Permizer.

To the best of our knowledge, Permizer is the first scheme to automatically manage permissions for mobile apps according to various user privacy preferences. The remainder of this article is organized as follows. Section II introduces previous related works. Section III formalizes the research problem. Section IV presents the problem-solving model in detail, and Section V presents the experimental results, evaluates the model in real-world scenarios and discusses our findings. Section VI concludes our work.

II. Background and Related Work

The research on automatic application permissions management can be categorized into three perspectives: 1) historical habits around privacy settings; 2) risk minimization; and 3) permission-function mapping.

A. Historical Settings-Based Permission Management

The most intuitive solution to achieve automatic permission management and relieve the user from the burden of setting permissions manually is to leverage the historical settings. More specifically, the assumption is that, if we know whether the user is privacy permissive or privacy conservative via users' historical settings, we can automatically configure permissions in accordance with this behavior. This type of method depends on knowing the user's privacy preference accurately [16].

Methodologies to extract privacy preferences fall into two categories: 1) investigating user privacy attitudes through questionnaires and 2) learning user privacy attitudes through their previous permission setting actions. For instance, Smarper [9] conducted a survey on 41 users that gathered 8521 privacy decisions, then employed a Bayesian linear regression approach to train a privacy decision prediction model. In contrast, Oglaza *et al.* [8] gathered privacy preference by collecting users' previous behaviors of granting or denying application permissions.

It is easy to see that such a solution is vulnerable since it heavily relies on the accurate extraction of privacy preferences. For example, an improper answer to the questionnaire by the user or insecure historic permission management behaviors would inevitably cause untrustworthy permission management. In addition, compared with our method, it is not flexible because this method gives users fewer choices to change their privacy preferences. In addition, this type of method does not benefit either privacy protection or functionality preservation.

B. Risk Minimization-Based Permission Management

To overcome the drawback of historic permission setting methods, which cannot guarantee privacy protection when configuring permissions, studies, including [10], [11] provide insight for users about permission decisions from the privacy risk perspective. Instead of gathering historical settings, models from this perspective will detect overly risky permissions and declare them to users; in this way, risk minimization-based models eliminate the dependence on user behavior. More specifically, Wang *et al.* [10] analyzed the risk of a single permission and a group of collaborative permissions, then selected a dangerous permission subset. Chitkara *et al.* [11] proposed a framework named ProtectMyPrivacy to infer the purpose of permission access, for instance, whether the permission is requested by a third-party library or the app itself for the sake of functionality, which will assist users in deciding whether certain permissions should be granted or not.

The methods above consider privacy protection when configuring permissions but fail to link a single permission to a specific app's functionality, thereby producing a biased solution for certain apps. In the worst cases, they ban important

TABLE I
DANGEROUS PERMISSIONS

GROUP	PERMISSION	EXPLANATION
CALENDAR	READ_CALENDAR WRITE_CALENDAR	*Allows an application to read/write the user's calendar data*
CALL_LOG	READ_CALL_LOG WRITE_CALL_LOG	*Allows an application to read/write the user's call log data*
CAMERA	CAMERA	*Allows an application to access the camera device*
CONTACTS	READ_CONTACTS WRITE_CONTACTS	*Allows an application to read/write the user's contacts data*
LOCATION	ACCESS_COARSE_LOCATION ACCESS_FINE_LOCATION	*Allows an app to access approximate/precise location*
MESSAGE	SEND_SMS/RECEIVE_SMS READ_SMS/WRITE_SMS RECEIVE_WAP_PUSH RECEIVE_MMS BROADCAST_WAP_PUSH BROADCAST_SMS	*Allows an application to send/receive/read/write/broadcast short messages/ multimedia messages/messages with hyperlinks*
MICROPHONE	RECORD_AUDIO	*Allows an application to record audio*
PHONE	PROCESS_OUTGOING_CALLS READ_PHONE_STATE ADD_VOICEMAIL CALL_PHONE GET_ACCOUNTS	*Allows an application to see the number being dialed; Allows to read phone state; Allows an application to add voicemails into the system; Allows the app to answer an incoming phone call; Allows the app to access to the list of accounts in the accounts service*
STORAGE	READ_EXTERNAL_STORAGE WRITE_EXTERNAL_STORAGE	*Allows an application to read/write from external storage*

permissions that support the given app's basic functions. In addition, the alerts caused by false positives diminish the user experience.

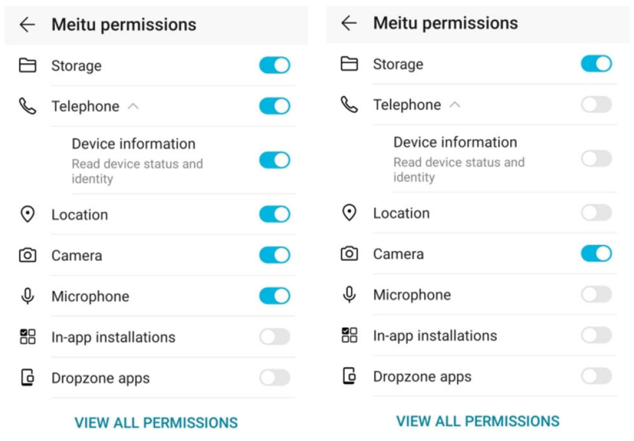
C. Permission-Function Mapping Scheme

When users adopt privacy-oriented permission management, the app's core functionality should not be jeopardized. For this reason, many works have proposed permission management schemes based on permission-function mapping. Permission-function mapping is used to evaluate how relevant a permission is to the app operation.

WHYPER [12] accomplished the first attempt to identify whether permission usage is stated in app description text. Qu *et al.* [13] developed AutoCog to capture the relatedness of permissions and functionality description, defined as *description-to-permission fidelity*. Instead of harnessing API documents as WHYPER did, AutoCog evaluated relatedness with a model trained on a large data set of app descriptions by applying a semantic extraction approach. Models that work in a similar fashion, such as AutoPer developed by Gao *et al.* [14] can automatically recommend users permissions decisions at the run time.

Huang *et al.* [15] combined privacy risk mitigation and permission-function mapping. Based on the assumption that similar applications ought to have similar permission configurations, they compared the permission requesting behaviors of popular apps and malware apps, then identified the abnormal permissions and removed them.

The above studies certainly contributed greatly to the research of mapping functions and descriptions, but few have presented the tradeoff between privacy risks and application functions or solutions that adjust to unique user privacy attitudes. Thus, many proposed methods cannot satisfy the personalized needs of different users, or even different preferences from the same user.


(a) Meitu permissions

Storage	●
Telephone	●
Device information — Read device status and identity	●
Location	●
Camera	●
Microphone	●
In-app installations	○
Dropzone apps	○

VIEW ALL PERMISSIONS

(b) Meitu permissions

Storage	●
Telephone	○
Device information — Read device status and identity	○
Location	○
Camera	●
Microphone	○
In-app installations	○
Dropzone apps	○

VIEW ALL PERMISSIONS

Fig. 1. Smartly reset the requested permissions in the example of Meitu. (a) Original requested permissions. (b) Reset permissions.

All the above works achieved various permission recommendation solutions. Enlightened by their methods and conclusions, we present Permizer, a one-button service to balance privacy with the function that is customized to individuals.

III. Problem Formalization

In this study, we focus on permissions in the Android ecosystem, including a total of nine categories with 32 permissions. Due to the unavailability of some permissions in the data set used in this study, we conduct our analysis on 25 dangerous permissions. Their corresponding explanations are shown in Table I.

Fig. 1 demonstrates the intention of our study. Take Meitu, a popular Chinese photography app, as an example, Fig. 1(a) introduces the original permissions Meitu requests from users when the app is installed and opened for the first time.

Clearly, unnecessary permissions are requested for a photography app, especially for privacy-conservative users. They may only wish to utilize the *STORAGE* and *CAMERA* permissions to operate the basic functionalities, as Fig. 1(b) shows. In contrast, privacy-permissive users may grant the *LOCATION* permission to add a watermark into the photograph, but these permissions may be vulnerable to an attack. With different user privacy preferences, the permission settings could be different. Therefore, the proposed framework is expected to automatically grant permissions based on the user's unique privacy preference to optimize both risks and function.

Given a corpus of applications $\mathcal{A}$, for application $a \in \mathcal{A}$. When a is downloaded from app store, we will be informed of its permission requesting list and function description. The dangerous permission set will be denoted as $\mathcal{P}$, for $\mathcal{P} = \{p_1, \ldots, p_n\}$. Here, n represents the number of dangerous permissions and equalizes 25 in this study. Requested dangerous permission set is denoted as $\mathcal{Q}$, for $\mathcal{Q} = \{q_1, \ldots, q_k\}$. Here, k represents the number of requested dangerous permissions. For dangerous permission set $\mathcal{P}$ we denote the corresponding risk vector as $\mathcal{R}$ where $\mathcal{R} = \{r_1, \ldots, r_n\}$. Similarly, functionality relatedness vector is $\mathcal{S}$ where $\mathcal{S} = \{s_1, \ldots, s_n\}$. α is the privacy preference factor that measures how much a user values his privacy, and ranges from 0 to 1. A more specific explanation of α is provided in Section IV. After having the above factors properly evaluated, the ideal model is expected to generate a content result, a binary decision vector $\mathcal{X} = \{x_1, \ldots, x_k\}$, whose element indicates 1 when the permission is recommended to be granted, 0 otherwise.

To determine which permissions should be granted based on their risk value and relatedness to functionality, the following three challenges must be solved.

1) How do we quantify risk $r \in \mathcal{R}$ of an individual permission if it has been granted?
2) How do we evaluate the relatedness $s \in \mathcal{S}$ of a single permission to a certain App?
3) What is the ideal model that takes three factors [privacy risk, *privacy preference* (α) and functionality] into account and captures the tradeoff?

IV. Methodology

Fig. 2 gives an architectural overview of Permizer. In this section, we explain the details of three modules included in Permizer, which are designed to address the three aforementioned challenges. The *risk estimation module* quantifies the risk of the single permission as well as permission combination as a solution to *Challenge* 1. The *permission-function mapping module* solves *Challenge* 2 by linking permissions with app functionality and measures the necessity of each permission with NLP techniques. The *permission assignment module* explains how the designed model captures the tradeoff between privacy and function with a multiobjective optimization model as a solution to *Challenge* 3.

A. Privacy Risk Estimation

In this module, Permizer evaluates the risk of granting dangerous permissions and explores how user privacy leakage

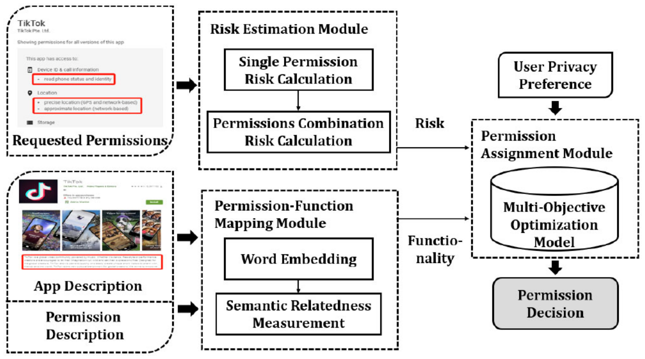

TikTok
Requested Permissions
App Description
Permission Description
Risk Estimation Module
Single Permission Risk Calculation
Permissions Combination Risk Calculation
Permission-Function Mapping Module
Word Embedding
Semantic Relatedness Measurement
Risk
Functionality
User Privacy Preference
Permission Assignment Module
Multi-Objective Optimization Model
Permission Decision

Fig. 2. Framework overview.

risks vary by adopting different permission settings. After an application has been installed, several basic permissions are required to guarantee the normal operation, such as *INTERNET* and *CHANGE_WIFI_STATE*. Recently developed apps may require additional permissions to provide its various services, such as *CAMERA and READ_CONTACT*, which fall under the category of dangerous permissions. Users have the choice to block the permission request if he/she finds that it generates a serious potential risk or makes a limited contribution to the app functionality.

It is relatively difficult to quantify the privacy leakage risk of permissions because different users may not share the same understanding of these permissions. However, the risk evaluation function ought to agree with the following three principles.

1) Malicious apps always have higher risks when granted permissions than benign ones.
2) The more frequently permissions are requested by benign apps, the less risky they are, and permissions requested frequently by malicious apps are more hazardous [17].
3) The more permissions the users grant, the higher risk they take.

Therefore, three steps are required to explore the risk function of permission settings. First, it is important to determine whether this application is benign or malware. In the real world, however, discernable malware applications are rare because the majority of users install only the most popular applications in the application store. Therefore, instead of categorizing the applications into benign and malware applications, we would rather assign a probability that it belongs to the benign or malware application group, which indicates the risk level of this application. This step will be introduced in part 1), and classic machine learning models will be employed to classify the apps. Second, we will quantify the single permission risk of different categories. Based on *principle* 2, an abnormal permission request is expected to be assigned a higher risk value, so there is a need to distinguish abnormal requests from benign and malware applications, and we will measure the permission risk using 2 proposed formulas in part 2). Finally, given the outcomes from 1) and 2), the risk from permission settings of certain applications can be computed in part 3). The required data set will be introduced as follows.

TABLE II
COMBINED DATA SET COMPOSITION

Dataset	Benign	Malware	Number of Dangerous Permission
Drebin	9083	2270	
AMD	600	1007	25
Combined	9683	3277	

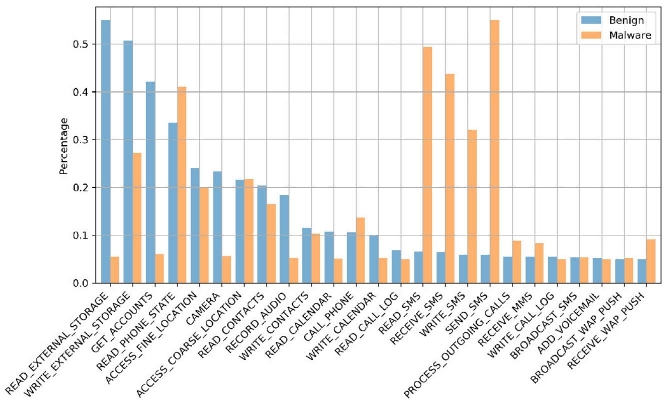


Fig. 3. Occurrence frequency of 25 dangerous permissions of benign and malware applications.

Data Description: We adopted the combined data set of part of the Drebin [18] and AMD data sets [19], in which the Drebin data set includes permission requesting list of 9083 benign apps and 2270 malware and AMD data set includes 600 benign apps and 1007 malware after preliminary processing. The reason that we use the combined data set is that applications in China are prone to overly request permissions, but apps in Drebin or AMD data set request much less and result in weak representativeness in the context of China application study. To address this problem, we screened all the apps with relatively longer requesting lists. Table II briefly describes the composition of the combined data set.

1) Risk Estimation on Application Level: Fig. 3 depicts the requesting frequency of 25 dangerous permissions. One obvious phenomenon is that permissions related to SMS are requested more frequently by malware applications.

Classifier Selection: In order to obtain the probability of a given application being benign or not, we use a binary classifier to give this assessment. Specifically, based on 25 dangerous permissions, we employ six classification algorithms: 1) logistic regression (LR); 2) naive Bayes classifier (NB); 3) K neighbors classifier (KNN); 4) decision tree (DT); 5) random forest (RF); and 6) AdaBoost (ADB). Eventually, DT and RF are selected due to their better performances in accuracy prediction. Table III displays the performances of selected classifiers. Apparently, DT and RF outperformed the rest of the classifiers. Due to the equivalent performances of DT and RF, and the purpose of validating the model's robustness, an external data set of malware applications was introduced to test both models. This data set is extracted from computer and security data set,[1] containing 667 malware apps and corresponding permission request lists. RF achieves an accuracy

[1]Mahindru, Arvind (2020), "Android permissions data set," Mendeley Data, V2, doi: 10.17632/b4mxg7ydb7.2

TABLE III
PERFORMANCE EVALUATION OF DIFFERENT CLASSIFIERS

Algorithm	Precision	Recall	F1 score
LR	0.88	0.90	0.89
NB	0.76	0.91	0.83
KNN	0.87	0.92	0.89
DT	0.89	0.92	0.91
RF	0.89	0.92	0.91
ADB	0.87	0.89	0.88

of 92% on this external data set, which made it the chosen algorithm for our study.

To date, a simple classification model with 25 dangerous permission as features has been completed to measure an individual application's general risk. When encountering a new application in the app store a_i, we would input the permission requesting list $\mathcal{Q}_i$ to the classification model $\mathcal{C}$ so that the risk value of this application $\text{Risk}(a_i) \in [0, 1]$ can be calculated with (1), which satisfies principle 1 malware always have higher risk to grant permissions than benign ones

$$\text{Risk}(a_i) = \text{Prob}\{a_i \in \mathcal{M}\} = \mathcal{C}(\mathcal{Q}_i) \tag{1}$$

where $\mathcal{M}$ is the malware set.

2) Risk Estimation on Permission Level: It is imperative for users to recognize that some permissions are relatively riskier than others, therefore, they should consider carefully when granting these permissions or simply deny them if they are not necessary. Aiming at escaping privacy risk as much as possible, privacy risk for each dangerous permission is then computed. However, what makes this measurement difficult is that users have different understandings of privacy risk. Certain permissions, such as CAMERA may be an important and private permission to one user, but not to another. Hence, to measure permissions risk objectively and impersonally, we proposed the following approach. Intuitively, applications with larger probabilities of being benign are less likely to leak user privacy information, that is, the risks of individual permissions from two categories are different and require independent calculation. Prior knowledge indicates that, given a large number of benign apps, the risk of permission j remains low if permission j is requested very frequently, as this permission might provide basic or general functions, and *vice versa*. This is not the case with malware apps, where the most frequently requested permissions carry more risks. Based on this assumption, we designed the following approach to calculate the risk of permission j from apps belonging to opposite categories, which fulfills principle 2. The more frequently permissions are requested by benign apps, the less risky they are

$$\begin{aligned} \text{Risk}\left\{p_j|a_i^b\right\} &= \text{MinMaxScaler}\left(-\ \text{Prob}\left\{p_j|a_i \in \mathcal{B}\right\}\right) \\ \text{Risk}\left\{p_j|a_i^m\right\} &= \text{MinMaxScaler}\left(\ \text{Prob}\left\{p_j|a_i \in \mathcal{M}\right\}\right) \end{aligned} \tag{2}$$

where $a_i \in \mathcal{A}$ is an application in the data set, $\mathcal{B}$ is the benign set, and $\mathcal{M}$ is the malware set. *MinMaxScaler*$(\cdot)$ is a function that scales the input into the range of 0 and 1. When a permission is frequently requested by benign applications, the risk of this permission in the benign group is considered to be minor because of the "minus" operator in (2), *vice versa*. Having applied these two formulas to the combined data set

TABLE IV
RISK VALUE OF DANGEROUS PERMISSIONS

Dangerous Permissions	Benign Group	Malware Group	Mean Risk Value
SEND_SMS	0.983	1.000	0.992
READ_SMS	0.968	0.888	0.928
RECEIVE_SMS	0.972	0.774	0.873
WRITE_SMS	0.981	0.542	0.761
READ_PHONE_STATE	0.431	0.722	0.576
RECEIVE_WAP_PUSH	1.000	0.082	0.541
PROCESS_OUTGOING_CALLS	0.989	0.077	0.533
CALL_PHONE	0.887	0.173	0.530
RECEIVE_MMS	0.991	0.065	0.528
BROADCAST_WAP_PUSH	1.000	0.004	0.502
ACCESS_COARSE_LOCATION	0.668	0.334	0.501
BROADCAST_SMS	0.993	0.007	0.500
ADD_VOICEMAIL	0.994	0.000	0.497
WRITE_CALL_LOG	0.992	0.000	0.496
WRITE_CONTACTS	0.867	0.108	0.488
READ_CALL_LOG	0.962	0.000	0.481
READ_CONTACTS	0.691	0.229	0.460
ACCESS_FINE_LOCATION	0.620	0.296	0.458
WRITE_CALENDAR	0.902	0.005	0.453
READ_CALENDAR	0.886	0.003	0.444
RECORD_AUDIO	0.731	0.004	0.367
CAMERA	0.634	0.011	0.323
WRITE_EXTERNAL_STORAGE	0.086	0.446	0.266
GET_ACCOUNTS	0.258	0.021	0.140
READ_EXTERNAL_STORAGE	0.000	0.011	0.005

and obtaining the risk value of two categories. Table IV lists our results regarding the risk value range and mean value, which could be utilized in the integer decision model in part $\mathcal{C}$. We will explain these results with several example permissions as follows.

1) *SEND_SMS* is a permission with 0.983 of risk value in the benign group and 1.000 in the malware group. It is requested very frequently by malware (over 50% shown in Fig. 3), while is requested at a lower probability by benign apps (less than 10% in Fig. 3). Hence, no matter this permission is requested by a benign app or a malicious app, its risk score remains high and ranges from 0.983 to 1.000, depending on the risk level of the app.
2) *READ_PHONE_STATE:* Differently from *SEND_SMS*, this permission is requested by both of the benign and malicious apps with a similar frequency. On the one hand, this permission presents a high risk (0.722 listed in Table IV) if requested by malicious apps. On the other hand, it is also considered relatively secure if requested by the benign apps. Therefore, its risk value will be regulated by the risk of the app, and the mean risk value is 0.576.
3) *READ_EXTERNAL_STORAGE* is rarely requested by malicious apps (less than 10% shown in Fig. 3) but often requested by benign apps (over 50% shown in Fig. 3). Therefore, it shows a low overall risk value (0.005 listed in Table IV).

In this way to compute risks, we avoid collecting personal or even biased privacy risk answers from questionnaires. In addition, this permission risk ranking result is highly consistent with the work of Wang, who employed three feature ranking techniques to explore permission-induced risk [10], which speaks to the correctness of our risk rating result.

3) Risk Estimation of Permissions Combination: The purpose of this step is to evaluate the risk of permission combinations when users alter permission settings and to formulate the ultimate risk function. To meet principle 3, the more permissions they grant, the higher risk they take. We propose the following method to evaluate the overall risk of permission configuration, which takes into account all the discussed factors above. We propose that the privacy leakage risk resulting from permission p_j from a certain app can be expressed as the formula of total probability. In reality, whether an app is benign or malicious is unknown to users, so given an app, we estimate permission risk Risk$\{p_j, a_i\}$ by multiplying conditional single permission risk with the likelihood of this app belonging to two categories, in which the likelihood is obtained in part 1). *PrivacyRisk*$\{\cdot\}$ represents overall risk evaluation and $\mathbf{x}$ denotes the binary decision vector of permission settings

$$\begin{aligned}\text{Risk}\{p_j, a_i\} &= \text{Risk}\left\{p_j|a_i^b\right\} * \text{Prob}\left\{a_i^b\right\}\\ &\quad + \text{Risk}\{p_j|a_i^m\} * \text{Prob}\{a_i^m\}\\ \text{PrivacyRisk}\{a_i, \mathbf{x}\} &= \sum_{j=1}^{n} \text{Risk}\{p_j, a_i\} * x_j. \end{aligned} \tag{3}$$

B. Mapping Single Permission With App's Functionality

In this section, we calculate the relevance of dangerous permissions and derive application functionality. Due to the poor performance of self-trained word embedding models caused by the limited crawled app description corpus, we adopt a widely used word embedding model trained on a large public corpora (e.g., Wikipedia or Twitter). First, we collect as many permission descriptions as possible, and then compute permission document vectors by employing the pretrained word embedding model Word2vec, which is introduced in parts 1) and 2). Second, multiple semantic distance calculations are compared and evaluated with regard to accuracy in part 3). All the above modules are packed into a semantic similarity computing model that is able to evaluate the relevance between the application functionality and its corresponding requested permissions. Part 4) concludes with the evaluation of the mapping model with a short Chinese video application.

1) Permission Description Text Collecting: The ideal permission description text ought to contain specific explanations of each requested permission, and more importantly, the usage statements. Precise explanations of dangerous permissions can be found in Android Developers, the official page of Android. Usage statements, however, have little valid source that summaries permissions usages in multitudinous apps. According to Jia *et al.* [20] and Chang *et al.* [21], a privacy policy is published to the users accompanied with the release of an app in the app market, stating that why this company acquires privacy data, what data they request to guarantee the normal running of this app and how they process and protect user personal data afterward. Namely, a legitimate privacy policy interprets permission usage in a particular app, which instructs us a straightforward and feasible way to collect sufficient permission usage text.

TABLE V
PERMISSION DESCRIPTION TEXT EXAMPLE

Permission	Description Text
READ_CALENDAR	**Allows an application to read the user's calendar data.** *read the personal schedule information entered by the user in the calendar; study plan, travel schedule, ticket booking, birthday reminder, meeting arrangement*
CAMERA	**Required to be able to access the camera device.** *take photos or record videos; scan QR codes and bar codes; share photos when shopping or posting comments; live or broadcast videos; facial recognition, intelligent recognition, users' feedback, profile pictures, visual reality*

Multitudinous apps in Google Play Store were categorized into 11 classes based on their functionality, such as shopping, communication, map, etc., 3 or 4 privacy policies are extracted from each class, which either belong to the most popular apps or perfectly elaborate all the items on the document. Having enumerated all 11 classes of privacy policies, we obtained a permission description corpus. Table V lists some examples of the work.

2) Universal Word Embedding: Word embedding is a fundamental tool in NLP using for mapping words to low dimensional vectors [22]. There are multiple methods to accomplish word embedding, and the most widely applied ones are word vector models because they have better performances in capturing the semantic meaning of the text [23]–[25]. Word2Vec [26] is a set of models that optimize their objective function over word co-occurrence within a window. Popular neural methods, such as Continuous Bag of Word (CBOW) or skip-gram with negative sampling (SGNS) are used to compute word distributional representations. The pretrained Word2Vec embedding we used in this article is google news vectors with the dimension of 300.[2] We then perform Word2vec on our permission description text in the following steps.

1) Preprocess the test set, including split the words and remove stop words.
2) Obtain the permission document vectors by getting the mean word vectors within a short document.
3) Reduce the dimension of document embedding vectors and project the document vectors into a 2-D vector space with t-SNE (a dimensionality reduction technique for high dimensional data projecting to 2 or 3 dimensions) to fulfill the visualization. Fig. 4 is the visualization of permission document vectors projecting to a 2-D space, where X and Y denote the 2-D reduced from high dimensions of word vectors.

We find that document similarities calculated with the above steps are consistent with human perception and present a good clustering performance. As is shown in Fig. 4, all dangerous permissions under the same category cluster together, because their semantics are closer. *ADD_VOICEMAIL* is an exception, although it belongs to the *PHONE* class, it is recorded that *allows apps to send voicemails to the system or add voice messages to the phone* in the textual description. Thus, it has a closer semantic distance to *RECORD_AUDIO* and SMS related permissions.

3) Semantic Distance Selection: Semantic distance calculation method is critical for obtaining an accurate semantic

[2]https://code.google.com/archive/

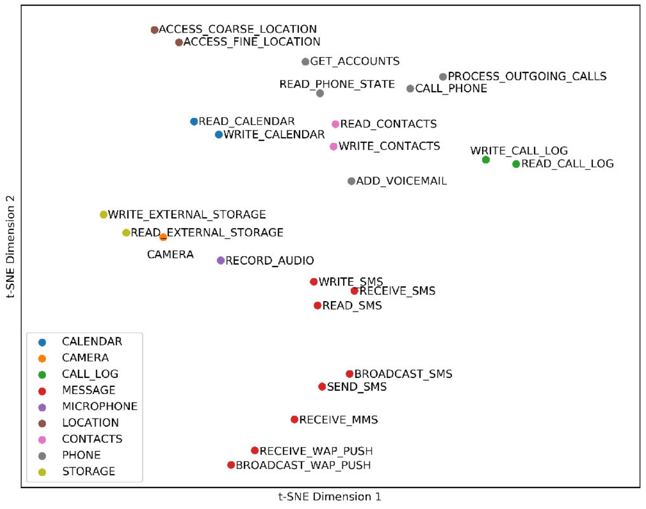


Fig. 4. Visualization of permission description text projection.

relatedness of permission description text and app function description text. Suppose μ and ν are sentence vectors, diverse distance methods work distinctively.

Euclidean Distance: It describes a straight-line distance between 2 points. The minimum similarity is 0 and the maximum is $+\infty$

$$d(\mu, \nu) = \|\mu - \nu\|_2. \quad (4)$$

Cosine similarity describes not merely distance but also the angle of 2 vectors. The minimum similarity is -1 and the maximum is $+1$ [27]

$$\cos(\mu, \nu) = \frac{\mu * \nu}{\|\mu\| * \|\nu\|} = \frac{\sum \mu_i \nu_i}{\sqrt{\sum \mu_i^2}\sqrt{\sum \nu_i^2}}. \quad (5)$$

Word mover's distance measures the minimum distance that the words in one document need to travel in semantic space to reach the words in the other document. In (6), **w, w**$'$ are the word vectors from two sequences with the length of n and n' [28]

$$\min \sum_{i,j} \gamma_{i,j} \left\| \mathbf{w}_i - \mathbf{w}'_j \right\|$$
$$\text{s.t.} \sum_j \gamma_{i,j} = \frac{1}{n}, \sum_i \gamma_{i,j} = \frac{1}{n'}. \quad (6)$$

Word rotator's distance is an improved method of Word mover's distance. It measures the minimum cosine distance

TABLE VI
PERMISSION RELATEDNESS MEASURED BY COSINE SIMILARITY

PERMISSION	SIMILARITY	PERMISSION	SIMILARITY
CAMERA	1.000	READ_EXTERNAL_STORAGE	0.425
RECORD_AUDIO	0.803	WRITE_CONTACTS	0.399
ACCESS_COARSE_LOCATION	0.670	WRITE_CALENDAR	0.398
ACCESS_FINE_LOCATION	0.669	RECEIVE_SMS	0.378
READ_PHONE_STATE	0.561	READ_CALENDAR	0.365
WRITE_CALL_LOG	0.547	PROCESS_OUTGOING_CALLS	0.360
WRITE_EXTERNAL_STORAGE	0.506	RECEIVE_MMS	0.173
READ_CONTACTS	0.491	GET_ACCOUNTS	0.172
READ_CALL_LOG	0.483	BROADCAST_SMS	0.157
SEND_SMS	0.476	RECEIVE_WAP_PUSH	0.068
CALL_PHONE	0.442	BROADCAST_WAP_PUSH	0.029
READ_SMS	0.439	ADD_VOICEMAIL	0.000
WRITE_SMS	0.429		

that the words in one document need to rotate in semantic space to reach the words in the other document [29]

$$\min \sum_{i,j} \gamma_{i,j}\left(1-\frac{\mathbf{w}_i\mathbf{w}'_j}{\|\mathbf{w}_i\| * \left\|\mathbf{w}'_j\right\|}\right)$$

$$\text{s.t.} \sum_{j} \gamma_{i,j} = \frac{\|\mathbf{w}_i\|}{Z}, \sum_{i} \gamma_{i,j} = \frac{\left\|\mathbf{w}'_j\right\|}{Z'}$$

$$Z = \sum_{i=1}^{n}\|\mathbf{w}_i\|,\ Z' = \sum_{j=1}^{n'}\left\|\mathbf{w}'_j\right\|. \quad (7)$$

Among the four semantic distance calculations, the document vectors of app functionality text are obtained before we apply Euclidean distance and Cosine distance calculation. Differently, Word movers' distance and Word rotator's distance require the word vectors within a document. We will select the right semantic distance method along with evaluating the performance of this module in the next part.

4) Permission-Function Mapping Evaluation: Due to the subjectivity of permission-function mapping in users' recognition and the absence of permission recommendation standards of apps, it is difficult to provide an accurate method to evaluate how important a permission is to a certain app in terms of functionality. Fortunately, a regulation document[3] from the Chinese government recommends the most necessary permissions as the minimum permission set for each type of apps, for the purpose of confining apps overly requesting behaviors. This could be a valuable reference. We set up a performance index to refer to the performance of the mapping model, in which the index denotes the percentage of apps whose minimum permission set is covered by the top k most related permissions computed from our model. For instance, given ten apps, we then compute top10 most related permissions for each app. If minimum permission sets of eight apps appear in the top ten permissions, the index will be 80%. Fig. 5 displays the comparison of the performance index when four types of similarity methods are adopted. As it can be seen, the Word rotator's distance outperforms the other three. Thus, the mapping model calculates the semantic similarity with Word rotator's distance.

[3]Information security technology—Basic specification for collecting personal information in mobile Internet applications.

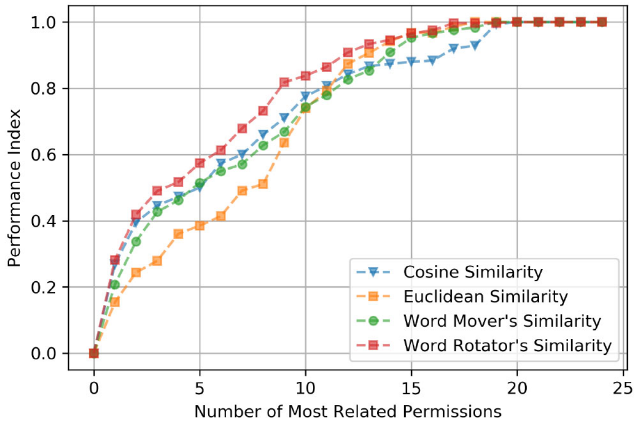


Fig. 5. Comparison of performance index of three word similarity calculation and the performance improvement as the number of most related permissions increases.

Table VI shows the permission-functionality relatedness of a video app measured by Word rotator's distance calculation. In this result, permission *CAMERA* and *RECORD_AUDIO* are highly related to the function of a video app. Location related permissions are slightly less related to this app for them bearing a specific function to recommend videos with other users in the same city, but this may not be what every user will consider in the first place.

To date, we built a semantic relatedness evaluation module composed of the above contents. Thus, permission relatedness to app functionality, denoted as Similarity$\{p, a\}$, will be calculated with the input of the functional description text into this model. For application a_i, we have

$$\text{Functionality}\{x_j, a_i\} = \sum_{j=0}^{n} \text{Similarity}\{p_j, a_i\} * x_j. \quad (8)$$

C. Multiobjective Optimization Modeling

Having obtained the variables of privacy risk and permission relatedness in the above study, we now propose an integer optimization model to leverage those two factors when users reach permission decisions.

Privacy Risk Minimization: For every application, two factors are contributing to the variation of privacy risk generated by using it, which are the risk of single permission and the permission setting. In objective function (9), $\mathbf{r}$ denotes permission risking vector from application a_i and $r_j \in (0, 1)$. $\mathbf{x}$ denotes binary decision vector which indicates user permission management, $x_j \in \{0, 1\}$

$$\min_{\mathbf{x}} \ \mathbf{r}'\mathbf{x} \tag{9}$$

where $\mathbf{r}'$ is the transpose of vector $\mathbf{r}$.

Functionality Maximization: Similarly, two factors are contributing to the variation of applications' functionality, which are the relatedness of single permission to function and the permission settings. In objective function (10), $\mathbf{s}$ denotes the permission-function relatedness vector from application i and $s_j \in (0, 1)$

$$\max_{\mathbf{x}} \ \mathbf{s}'\mathbf{x}. \tag{10}$$

Leverage Objective Function: α is the weight of privacy risk and denotes users' privacy preference. It measures privacy attitudes in this study, various to different users. In (11), α is the weight of the risk subobjective function to mediate the relation of privacy risk and app functionality. Users can change the permission recommendation result by adjusting this parameter. For example, a privacy-conservative user may value her privacy so much that she is allowed to grant critical permissions only. Therefore, he will then adopt a small α. But a privacy-permissive user may value the app functionality much more than his privacy concern, on which occasion a larger α is suitable to him. $\alpha \in (0, 1)$

$$\arg\min_{\mathbf{x}} \ \alpha * \mathbf{r}'\mathbf{x} - (1 - \alpha) * \mathbf{s}'\mathbf{x}. \tag{11}$$

The feasible region of this model is now unconfined, which means the optimal solution hides among $2^n = 2^{25}$ feasible solutions (25 dangerous permissions and two options toward each). Therefore, other rules and restrictions have to be designed to confine the feasible region and improve the solution seeking efficiency.

Constraint 1: Set the permission decision variable 0 when it is not in requested permission set Q. In the real scenario, the majority of apps, especially benign ones would not request all dangerous permissions. It can be observed that most apps request no more than 20 permissions, which provides us a perspective to reduce the dimension of the permission decision vector

$$\begin{gathered}\mathbf{x}^{\mathcal{Q}} = \left(x_1, \ldots, x_j, \ldots, x_n\right) \\ x_j = \begin{cases} 0, & \text{if } p_j \notin \mathcal{Q} \\ 1, & \text{otherwise} \end{cases} \\ \text{Finally,} \quad \mathbf{x} \preceq \mathbf{x}^{\mathcal{Q}}. \end{gathered} \tag{12}$$

Scale the Elements of Risk Vector and Similarity Vector: After setting $\mathbf{x} \preceq \mathbf{x}^{\mathcal{Q}}$, the sum of the risk vector element and similarity element will be altered. To ensure preciseness and rigorousness, we expect to keep the ultimate risk value and function value to 1 if the user accepts all the permission requests from certain application. Therefore, elements of the risk vector and similarity vector are scaled with the following constraints. If $\mathbf{x} = \mathbf{x}^{\mathcal{Q}}$

$$\mathbf{r}'\mathbf{x} = 1, \ \ \mathbf{s}'\mathbf{x} = 1. \tag{13}$$

Constraint 2: Set the permission decision variable as 1 when this permission is the most important permission to the app. Though the previously mentioned Chinese document provides a fundamental set as a reference, it contains limited categories and is not customary to each app. It is then challenging to define a fundamental permission set for the reason that every user has a unique understanding of fundamental function even on the same application. To act prudently, we decide to grant the most related permission as the last constraint

$$\begin{gathered}\mathbf{x}^{\mathcal{C}} = \left(x_1, \ldots, x_j, \ldots, x_n\right) \\ x_j = \begin{cases} 1, & \text{if } x_j = \max(\mathbf{s}) \\ 0, & \text{otherwise.} \end{cases}\end{gathered} \tag{14}$$

Given all definitions we need, finally, we have our optimization framework, note that (11) can be rewritten as $\alpha * (\mathbf{r}' - \mathbf{s}')\mathbf{x} - \mathbf{s}'\mathbf{x}$, so our optimization model is

$$\begin{aligned} \arg\min_{\mathbf{x}} \quad & \alpha * \left(\mathbf{r}' - \mathbf{s}'\right)\mathbf{x} - \mathbf{s}'\mathbf{x} \\ \text{s.t.} \quad & \mathbf{x} + \boldsymbol{\rho} \preceq \mathbf{x}^{\mathcal{Q}} \\ & \mathbf{x} - \boldsymbol{\tau} \preceq \mathbf{x}^{\mathcal{C}} \\ & \boldsymbol{\rho} \succeq \mathbf{0}, \boldsymbol{\tau} \succeq \mathbf{0} \\ & x_j \in \{0, 1\} \end{aligned} \tag{15}$$

where $\boldsymbol{\rho}$ and $\boldsymbol{\tau}$ are the slack variable and surplus variable, respectively. As we can see, the above optimization is a zero-one or binary integer programming problem (ZOIP) [30]. Unlike linear programming problems, integer programming problems are very difficult to solve for the absence of the efficient general algorithm. Fortunately, after adding two constraints given above, the search space is relatively small, and we can find the exact solution of this ZOIP problem with some off-the-shelf tools. In practical, we use Python Library PuLP[4] to solve this optimization problem.

V. Evaluation and Discussion

To prove the effectiveness of our permission decision model, we implement it on a set of apps for evaluation and discussion. In this section, we focus on the analysis of how user privacy preference α influences the optimal solution of Permizer with examples, based on which it will provide practical guidance on the user permission management. First, we start by introducing a real-world data set, then discuss the meaning and variation of risk value and function value when privacy preferences change. We divide α into three intervals. Finally, model evaluation is accomplished in each interval by exhibiting the model outcomes for an IoT app and a video app.

Data Set: We crawl app description and their permission description texts from Google Appstore. The experiment set contains 58 most popular applications that have at least ten million downloads from 11 categories. Details are displayed in Table VII. As one can see, the mean of requested permission number is over 7 and exceeds 12 in the IoT category, which proves that permission over-privilege is not an individual behavior but a common phenomenon.

[4]https://pypi.org/project/PuLP/

TABLE VII
DATA SET DESCRIPTION

Category	Social	Finance	Life	Music	News	Photo	Shopping	Map	Video	Travel	IoT
Number	5	5	6	5	5	5	5	5	7	5	5
Mean Requested Permissions	9.2	11.2	9.5	9.2	8.8	7.2	9.4	8.4	10.6	11	12.4

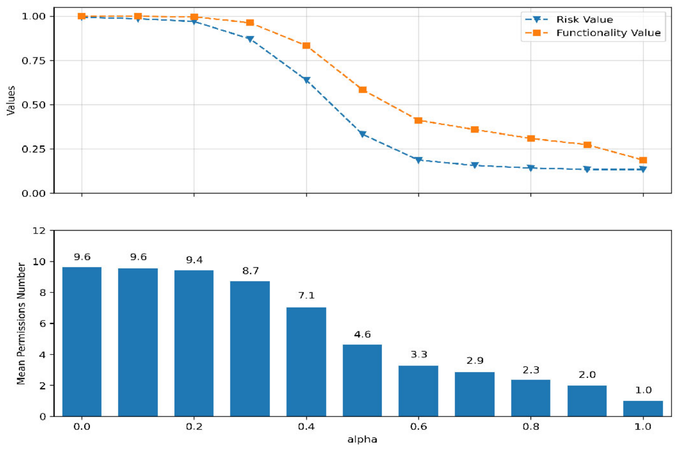
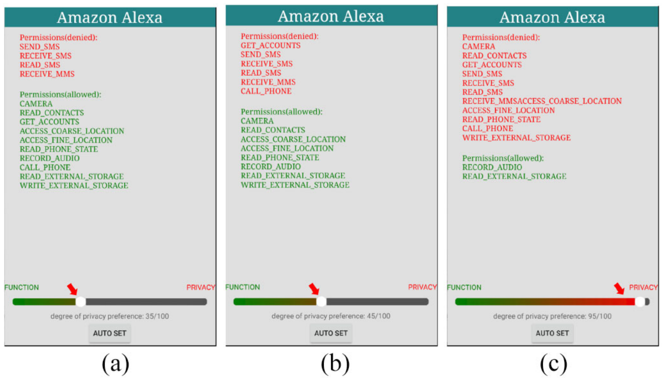


Fig. 6. Trend of risk value, function value, and mean recommended permission number as alpha changes.


(a)

Amazon Alexa

Permissions(denied):
SEND_SMS
RECEIVE_SMS
READ_SMS
RECEIVE_MMS

Permissions(allowed):
CAMERA
READ_CONTACTS
GET_ACCOUNTS
ACCESS_COARSE_LOCATION
ACCESS_FINE_LOCATION
READ_PHONE_STATE
RECORD_AUDIO
CALL_PHONE
READ_EXTERNAL_STORAGE
WRITE_EXTERNAL_STORAGE

degree of privacy preference: 35/100

(b)

Amazon Alexa

Permissions(denied):
GET_ACCOUNTS
SEND_SMS
RECEIVE_SMS
READ_SMS
RECEIVE_MMS
CALL_PHONE

Permissions(allowed):
CAMERA
READ_CONTACTS
ACCESS_COARSE_LOCATION
ACCESS_FINE_LOCATION
READ_PHONE_STATE
RECORD_AUDIO
READ_EXTERNAL_STORAGE
WRITE_EXTERNAL_STORAGE

degree of privacy preference: 45/100

(c)

Amazon Alexa

Permissions(denied):
CAMERA
READ_CONTACTS
GET_ACCOUNTS
SEND_SMS
RECEIVE_SMS
READ_SMS
RECEIVE_MMSACCESS_COARSE_LOCATION
ACCESS_FINE_LOCATION
READ_PHONE_STATE
CALL_PHONE
WRITE_EXTERNAL_STORAGE

Permissions(allowed):
RECORD_AUDIO
READ_EXTERNAL_STORAGE

degree of privacy preference: 95/100

Fig. 7. Display of recommended permissions of Amazon Alexa from Permizer. (a) $\alpha = 0.35$. (b) $\alpha = 0.45$. (c) $\alpha = 0.95$.

A. *Influence of Privacy Preference on the Model*

Some users greatly value their privacy while others may have the completely opposite attitude. To cater to users with diverse privacy preferences, one parameter in the model, privacy preference, depends on the users.

Implemented Permizer with the above data set as privacy preference α changes, we calculate the objective functions respectively, and the mean recommended permission numbers for 58 apps. Given the case when all permissions are granted, the risk and the fully intact functionality are both scaled to 1. As is displayed in Fig. 6, when a user is unconcerned about his privacy and uses a small value for α, the result obtained from Permizer is that most of the permissions will be granted to retain full functionality. However, there are a small set of permissions that will not be granted as they are either highly risky or totally unrelated to the app function. As α gets larger, the permission decision model increasingly values privacy and decreases the permission number to minimize privacy risk. Within this interval, both objective values descend drastically, and Permizer recommends users to grant 3–7 permission to every application; when α approaches to 1, this model becomes quite cautious and only grants the most fundamental permissions that guarantee basic application function. The above analysis explains the descending trend of risk value, function value and the recommended permission number in Fig. 6. Based on the amplitude of variation in the figure, we decide to split α into three intervals, $0.00-0.35$, $0.35-0.60$, and $0.60-1.00$, each serving a certain type of user.

B. *Evaluation With "Amazon Alexa" and "TikTok"*

To illustrate the performance of Permizer, we present 2 case studies from different functional categories: 1) IoT and 2) Video. Amazon Alexa is an application that manages Alexa-enabled devices and it helps to control or check the status of smart home appliances. TikTok is one of the most popular Chinese short video applications known as Douyin. It helps users express themselves and record their lives. Besides recommending intriguing short videos intelligently, it also creates an atmosphere that users are promoted to increase socialization by sharing their videos online. Amazon Alexa and TikTok require 14 and 10 dangerous permissions, respectively. Apart from the following, eight permissions in common: 1) *CAMERA*; 2) *ACCESS_COARSE_LOCATION*; 3) *ACCESS_FINE_LOCATION*; 4) *READ_EXTERNAL_STORAGE*; 5) *READ_CONTACTS*; 6) *RECORD_AUDIO*; 7) *READ_PHONE_STATE*; and 8) *WRITE_EXTERNAL_STORAGE*, Alexa additionally requires six other permissions: 1) *GET_ACCOUNTS*; 2) *CALL_PHONE*; 3) *SEND_SMS*; 4) *READ_SMS*; 5) *RECEIVE_SMS*; and 6) *RECEIVE_MMS*. For TikTok, two other permissions are further required: 1) *WRITE_CALENDAR* and 2) *READ_CALENDAR*.

In the following, we select three representative α from these intervals, and each α can represent one privacy preference scenario. We perform Permizer on the real-world data set with the given α. To present more intuitively, the recommended permission decisions calculated by Permizer are listed under different privacy preference settings. Fig. 7 displays the interface of Permizer and recommended permissions for Amazon Alexa.

1) Lower Privacy Preference ($\alpha = 0.35$): In this scenario, our model regards functionality as the most important factor. As expected, it denies only the riskiest or unrelated permissions. Fig. 8 reveals that 98% percent of applications' functionality value is over 0.8, while their risk value is relatively low, and less than 30% of permissions are denied to

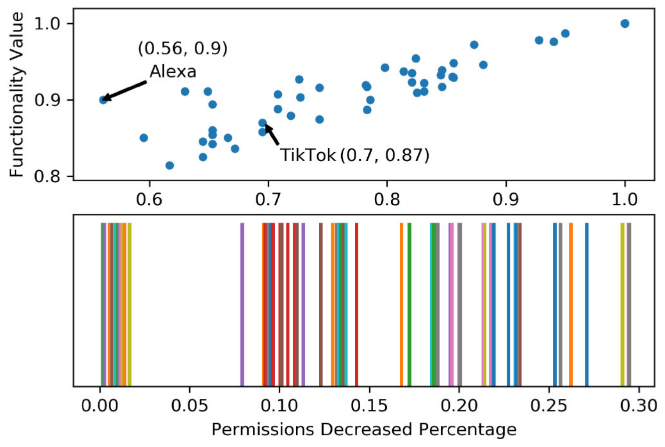


Fig. 8. Display of functionality value, risk value, and permissions decreased percentage when $\alpha = 0.35$.

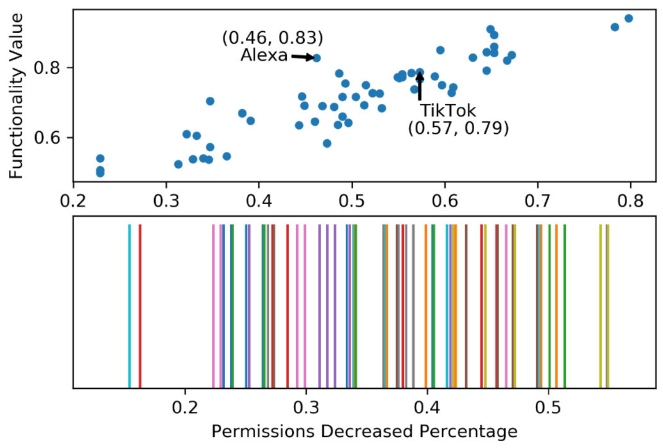


Fig. 9. Display of functionality value, risk value, and permissions decreased percentage when $\alpha = 0.45$.

minimize privacy risk. For example, the risk of Alexa's permission settings drops down to 0.56, and this permission setting results in a 10% loss in function. Permizer rejects four permissions: 1) *SEND_SMS*; 2) *READ_SMS*; 3) *RECEIVE_SMS*; and 4) *RECEIVE_MMS* from Alexa's original request list. This result is reasonable because permissions related to SMS/MMS are hazardous and do not enable major functions. Meanwhile, Permizer rejects SMS and MMS related permissions as well for TikTok, which leads to a 13% loss in function and a 30% decrease in risk. This scenario is designed to protect the privacy of users who have high privacy risk tolerance, prefer complete functionality and wish to avoid further efforts on permission configuration.

2) Moderate Privacy Preference ($\alpha = 0.45$): In this scenario, Permizer values both factors and attempts to balance them for the optimal solution. Fig. 9 reveals that functionality declines to 0.5–0.9 to avoid risk as much as possible. As a result, approximately 20%–50% of permissions are denied. The risk of the recommended permission setting for Alexa continues to drop in this case, and stabilizes at a level of 0.46 when alpha is 0.45, which comes at the price of 17% functional loss. Alexa is now not permitted to text people or automatically call phones through voice commands due to the rejection of related permissions. TikTok is assigned a functionality value equivalent to 0.79, which decreases privacy risk by 43%. Four permissions related to calendar and location access are denied by the Permizer in this scenario. Therefore, TikTok users will no longer be recommended videos near their location or receive unknown, trivial services related to calendar. Table VIII lists the recommended permissions for both apps. This scenario is suitable for users who are concerned about privacy risk but are reluctant to sacrifice functionality. Another detail to be noted is that in this privacy preference interval, objective function values fluctuate drastically with even minor changes of α. As a result, it is trickier for Permizer to capture users' privacy preferences accurately when compared to the other two intervals. One feasible solution is to learn users' historical permissions setting behaviors or ask users to establish several example permission settings first and observe their precise preferences.

TABLE VIII
RECOMMENDED PERMISSIONS OF ALEXA AND TIKTOK WHEN $\alpha = 0.45$

	Alexa
1	READ_PHONE_STATE
2	ACCESS_COARSE_LOCATION
3	READ_CONTACTS
4	ACCESS_FINE_LOCATION
5	RECORD_AUDIO
6	CAMERA
7	WRITE_EXTERNAL_STORAGE
8	READ_EXTERNAL_STORAGE
	TikTok
1	READ_PHONE_STATE
2	READ_CONTACTS
3	RECORD_AUDIO
4	CAMERA
5	WRITE_EXTERNAL_STORAGE
6	READ_EXTERNAL_STORAGE

3) Higher Privacy Preference ($\alpha = 0.95$): In this scenario, the privacy risk factor is a high priority. Our model confines this factor to the range of 0.1–0.45 at the price of 75%–95% functional loss, as shown in Fig. 10. Approximately 80% of permissions are denied for most apps in this experiment. Both Alexa and Tiktok are granted merely two or three permissions, listed in Table IX. Alexa now retains only two permissions, which contribute to 18% of the total functionalities. It can still use human voice commands to manage smart home devices and store necessary data, with the overall risk decreased to 0.05. Permizer decreases privacy risk for users of TikTok by 89% and maintains 24% of functionality. TikTok users can still view or make videos and also upload the videos to the app. Information as to whether people from their contact are using it is also available. Notably, the low functionality percentage in our study does not render the application unusable, instead, it guarantees the basic function of the application while discarding additional functions selectively. In the case of TikTok, the three recommended permissions still allow users to access the most important services, such as browsing and recording videos. This scenario could be utilized to manage permissions for users who are exceedingly careful about privacy protection and do not wish to be asked about permissions often.

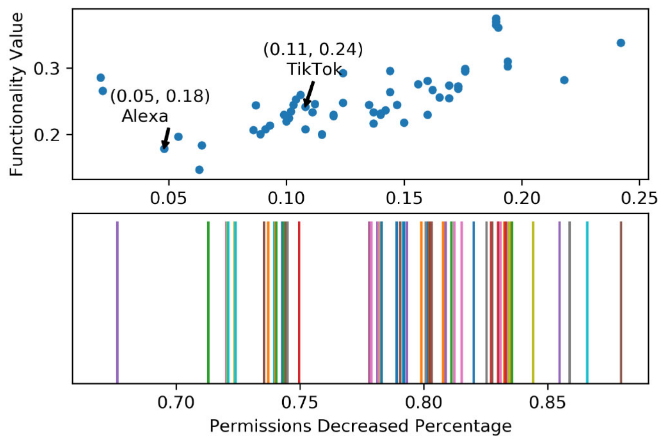


Fig. 10. Display of functionality value, risk value, and permissions decreased percentage when α = 0.95.

TABLE IX
RECOMMENDED PERMISSIONS WHEN α = 0.95

	Alexa
1	RECORD_AUDIO
2	READ_EXTERNAL_STORAGE
	TikTok
1	READ_CONTACTS
2	CAMERA
3	READ_EXTERNAL_STORAGE

VI. CONCLUSION

In this article, we presented a permission decision model to assist users to achieve a better management of permission requests from numerous IoT applications. This model was designed to leverage two critical factors that users concern primarily which are functionality and privacy risk, and realised an automatic permission management scheme. It contains three components: 1) risk evaluation; 2) functionality calculation; and 3) a multiobjective optimization model.

We further discussed the generality of the model by implementing it on a real-world data set and adjusting the critical parameter. Three scenarios were introduced based on two case studies of an IoT app named Amazon Alexa and a short video app named TikTok, to evaluate the proposed permission decision model and analyze its applicability. Results revealed that our model efficiently decreases privacy risk and is applicable for users with all types of privacy attitudes.

For future work, we plan to collect and adopt a more detailed and extensive data set with category information when conducting risk evaluation, because a function categorized data set will largely improve the accuracy of risk calculation. Moreover, we will improve the generality of the model by complementing permission preference factor.

REFERENCES

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions,” *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013.

[2] *The Mobile Economy*, GSMA, London, U.K. Accessed: Jun. 24, 2020. [Online]. Available: https://www.gsma.com/mobileeconomy/

[3] X. Zheng, Z. Cai, and Y. Li, “Data linkage in smart Internet of Things systems: A consideration from a privacy perspective,” *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 55–61, Sep. 2018.

[4] B. Camille. *Top Privacy Leaks From 2019*. Accessed: Mar. 6, 2020. [Online]. Available: https://berty.tech/blog/top-privacy-leak-2019

[5] Y. Liang, Z. Cai, J. Yu, Q. Han, and Y. Li, “Deep learning based inference of private information using embedded sensors in smart devices,” *IEEE Netw.*, vol. 32, no. 4, pp. 8–14, Jul./Aug. 2018.

[6] Z. Cai and X. Zheng, “A private and efficient mechanism for data uploading in smart cyber-physical systems,” *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 2, pp. 766–775, Apr.–Jun. 2020.

[7] L. Yu, X. Luo, X. Liu, and T. Zhang, “Can we trust the privacy policies of android apps?” in *Proc. 46th Annu. IEEE/IFIP Int. Conf. Depend. Syst. Netw. (DSN)*, Toulouse, France, 2016, pp. 538–549.

[8] A. Oglaza, R. Laborde, P. Zaraté, A. Benzekri, and F. Barrère, “A new approach for managing Android permissions: Learning users' preferences,” *EURASIP J. Inf. Security*, vol. 13, no. 1, pp. 1–16, 2017.

[9] K. Olejnik, I. Dacosta, J. S. Machado, K. Huguenin, M. E. Khan, and J.-P. Hubaux, “SmarPer: Context-aware and automatic runtime-permissions for mobile devices,” in *Proc. IEEE Symp. Security Privacy (SP)*, San Jose, CA, USA, 2017, pp. 1058–1076.

[10] W. Wang, X. Wang, D. Feng, J. Liu, Z. Han, and X. Zhang, “Exploring permission-induced risk in android applications for malicious application detection,” *IEEE Trans. Inf. Forensics Security*, vol. 9, pp. 1869–1882, 2014.

[11] S. Chitkara, N. Gothoskar, S. Harish, J. I. Hong, and Y. Agarwal, “Does this app really need my location?: Context-aware privacy management for smartphones,” *Proc. ACM Interact. Mobile Wearable Ubiquitous Technol.*, vol. 1, no. 3, pp. 1–22, 2017.

[12] R. Pandita, X. Xiao, W. Yang, W. Enck, and T. Xie, “WHYPER: Towards automating risk assessment of mobile applications,” in *Presented at the 22nd USENIX Security Symp. (USENIX Security)*, 2013, pp. 527–542.

[13] Z. Qu, V. Rastogi, X. Zhang, Y. Chen, T. Zhu, and Z. Chen, “AutoCog: Measuring the description-to-permission fidelity in android applications,” in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2014, pp. 1354–1365.

[14] H. Gao *et al.*, “AutoPer: Automatic recommender for runtime-permission in android applications,” in *Proc. IEEE 43rd Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, vol. 1. Milwaukee, WI, USA, 2019, pp. 107–116.

[15] K. Huang, J. Han, S. Chen, and Z. Feng, “A skewness-based framework for mobile app permission recommendation and risk evaluation,” in *Proc. Int. Conf. Serv. Orient. Comput.*, 2016, pp. 252–266.

[16] J. Lin, B. Liu, N. Sadeh, and J. I. Hong, “Modeling users' mobile app privacy preferences: Restoring usability in a sea of permission settings,” in *Proc. 10th Symp. Usable Privacy Security (SOUPS)*, 2014, pp. 199–212.

[17] A. Alshehri, P. Marcinek, A. Alzahrani, H. Alshahrani, and H. Fu, “PUREDRoid: Permission usage and risk estimation for android applications,” in *Proc. 3rd Int. Conf. Inf. Syst. Data Min.*, 2019, pp. 179–184.

[18] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, “Drebin: Effective and explainable detection of android malware in your pocket,” in *Proc. Netw. Distrib. Syst. Security Symp. (NDSS)*, vol. 14, 2014, pp. 23–26.

[19] M. Dimjašević, S. Atzeni, I. Ugrina, and Z. Rakamaric, “Evaluation of android malware detection based on system calls,” in *Proc. ACM Int. Workshop Security Privacy Anal.*, 2016, pp. 1–8.

[20] Q. Jia, L. Zhou, H. Li, R. Yang, S. Du, and H. Zhu, “Who leaks my privacy: Towards automatic and association detection with GDPR compliance,” in *Proc. Int. Conf. Wireless Algorithms Syst. Appl.*, 2019, pp. 137–148.

[21] C. Chang, H. Li, Y. Zhang, S. Du, H. Cao, and H. Zhu, “Automated and personalized privacy policy extraction under GDPR consideration,” in *Proc. Int. Conf. Wireless Algorithms Syst. Appl.*, 2019, pp. 43–54.

[22] J. Turian, L. Ratinov, and Y. Bengio, “Word representations: A simple and general method for semi-supervised learning,” in *Proc. 48th Annu. Meeting Assoc. Comput. Linguistics*, 2010, pp. 384–394.

[23] R. Socher, B. Huval, C. D. Manning, and A. Y. Ng, “Semantic compositionality through recursive matrix-vector spaces,” in *Proc. Joint Conf. Empirical Methods Nat. Lang. Process. Comput. Nat. Lang. Learn.*, 2012, pp. 1201–1211.

[24] M. Sahlgren, “The word-space model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces,” Ph.D. dissertation, Dept. Linguistics, Stockholm Univ., Stockholm, Sweden, 2006.

[25] M. Faruqui, J. Dodge, S. K. Jauhar, C. Dyer, E. Hovy, and N. A. Smith, “Retrofitting word vectors to semantic lexicons,” in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics Hum. Lang. Technol.*, 2015, pp. 1606–1615.

[26] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran, 2013, pp. 3111–3119.

[27] H. Dubossarsky, D. Weinshall, and E. Grossman, "Outta control: Laws of semantic change and inherent biases in word representation models," in *Proc. Conf. Empirical Methods Nat. Lang. Process.*, 2017, pp. 1136–1145.

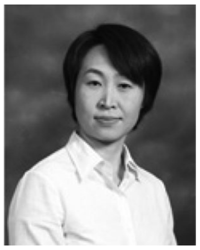
[28] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger, "From word embeddings to document distances," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 957–966.

[29] S. Yokoi, R. Takahashi, R. Akama, J. Suzuki, and K. Inui, "Word rotator's distance," 2020. [Online]. Available: https://arxiv.org/abs/2004.15003.

[30] D. Bertsimas and J. N. Tsitsiklis, *Introduction to Linear Optimization*. vol. 6. Belmont, MA, USA: Athena Sci., 1997.

Yiting Qu received the B.Sc. degree in management science and engineering from Shandong University, Jinan, China, in 2018. She is currently pursuing the M.S.E.M. degree from the Antai College of Economics and Management, Shanghai Jiao Tong University, Shanghai, China.

Her research interests include risk evaluation, privacy protection, and other management areas.

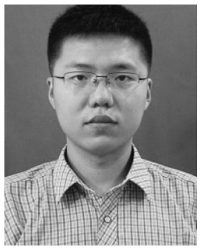


Suguo Du received the Ph.D. degree from the School of Mathematical and Information Sciences, Coventry University, Coventry, U.K., in 2002.

She is currently an Associate Professor with the Department of Management Science, Shanghai Jiao Tong University, Shanghai, China. Her research has been supported by the National Science Foundation of China. Her current research interests include risk and reliability assessment, vehicular networks security and privacy protection, and social networks security management.

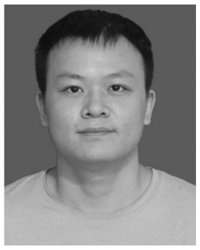
Shaofeng Li (Graduate Student Member, IEEE) is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China.

He focuses primarily on the areas of machine learning and security, specifically exploring the robustness of machine learning models against various adversarial attacks.



Yan Meng (Graduate Student Member, IEEE) received the B.S. degree in electronic and information engineering from Huazhong University of Science and Technology, Wuhan, China, in 2016. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China.

His research interests include wireless network security and IoT security.



Le Zhang (Graduate Student Member, IEEE) received the B.S. degree in computer science and technology from Shanghai Jiao Tong University, Shanghai, China, in 2020, where he is currently pursuing the postgraduation degree with the Department of Computer Science and Engineering.

His research interests include Ad fraud detection and wireless network privacy.

Haojin Zhu (Senior Member, IEEE) received the B.Sc. degree in computer science from Wuhan University, Wuhan, China, in 2002, the M.Sc. degree from Shanghai Jiao Tong University, Shanghai, China, in 2005, and the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2009.

Since 2017, he has been a Full Professor with the Computer Science Department, Shanghai Jiao Tong University. He has published more than 50 journals, including: IEEE Journal on Selected Areas in Communications, IEEE Transactions on Dependable and Secure Computing, IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Mobile Computing, IEEE Transactions on Information Forensics and Security, IEEE Transactions on Wireless Communications, and IEEE Transactions on Vehicular Technology, and more than 60 international conference papers, including IEEE S&P, ACM CCS, ACM MOBICOM, NDSS, ACM MOBIHOC, IEEE INFOCOM, and IEEE ICDCS. His current research interests include network security and privacy enhancing technologies.

Prof. Zhu received a number of awards including: the IEEE ComSoc Asia–Pacific Outstanding Young Researcher Award in 2014, the Top 100 Most Cited Chinese Papers Published in International Journals in 2014, the Supervisor of Shanghai Excellent Master Thesis Award in 2014, the Distinguished Member of the IEEE INFOCOM Technical Program Committee in 2015, the Outstanding Youth Post Expert Award for Shanghai Jiao Tong University in 2014, and the SMC Young Research Award of Shanghai Jiao Tong University in 2011. He was a co-recipient of best paper awards of IEEE ICC in 2007 and Chinacom in 2008, the IEEE GLOBECOM Best Paper Nomination in 2014, and the WASA Best Paper Runner-Up Award in 2017. He received the Young Scholar Award of Changjiang Scholar Program by Ministry of Education of China in 2016. He is a Member of ACM.