SmartPI: Understanding Permission Implications of Android Apps from User Reviews

Run Wang, Zhibo Wang, *Senior Member, IEEE*, Benxiao Tang, *Student Member, IEEE*, Lei Zhao, *Member, IEEE*, and Lina Wang, *Member, IEEE*

Abstract—With the unprecedented convenience brought by Apps on mobile devices, we are facing severe security attacks and privacy leakage caused by them since they may stealthily access unclaimed or unneeded permissions for some purposes. Many works strive to discover these malicious apps using program analysis techniques, however, they fail to tell users why an app needs to request the permission from users' perspective. In this paper, we leverage the power of the crowdsourced user reviews to understand why an app requests a permission. We propose a framework, called SmartPI, that automatically identifies functionality-relevant user reviews and infers the permission implication of them, bridging the gap between the functionalities and the actual behaviors of an app. In particular, we extract features from the platform documents to identify functionality-relevant user reviews from noisy crowdsourced user reviews with Natural Language Processing (NLP) techniques. The topic model is further adopted to infer the permission implications of apps from the functionality-relevant user reviews. More than 20,000 apps, 2,653,159 users, and 4,247,769 user reviews are crawled from Google Play as a real-world dataset to evaluate the performance of SmartPI. The experiments results show that the permission usage of apps can be better reflected by user reviews than the claimed descriptions of apps.

Index Terms—User reviews, permissions, overprivilege, malware, natural language processing

1 Introduction

Mobile devices such as smartphones and smartwatchs are becoming more and more popular in our daily lives. According to the public report published by GSMA [1], the number of mobile users is over 5 billion since the middle of 2017. The statistical data show that there are more than two million applications (a.k.a. apps) in both Google Play Store and Apple's App Store, and the number is still increasing rapidly every day. Reports also show that over 1.5 billion apps downloaded from Google Play every month. While enjoying the unprecedented convenience of mobile apps with rich features, we are facing severe security attacks and privacy leakage caused by unsafe mobile apps. Some researches pointed out that there are more than one third malwares declaring permissions beyond their claimed functionalities on Android platform. These apps request unneeded permissions for malicious intentions or carelessness and fail to follow the principle of least-privilege [2], which will bring some attacks to mobile devices such as privilege escalation, application collusion, confused deputy [3] and privacy leakage (e.g., a user's private data are read and sent to a remote server without attracting its attention).

Previous malware detection approaches like static analysis [4], [5] and dynamic analysis [6], [7] mainly focus on inspecting the behaviors performed by apps, but fail to check the functionalities claimed by apps. For example, a *hotel booking* app accesses users' locations and then send them to a trusted location-based server to provide the nearby hotel information. However, if it sends users' location information to other third parties, it would be considered as a malicious behavior with previous malware detection techniques. Thus, it is necessary to bridge the gap between the real behaviors and the functionalities claimed by the apps. Following this idea, some recent studies [8], [9], [10], [11] proposed understanding the permission request of apps from app descriptions. In their work, app descriptions are taken as kinds of users' expectations of apps [8] and they check whether an app's behavior is within user expectation. However, app descriptions can be easily crafted by developers to deceive users as well as detection systems. To solve this problem, in this paper, we leverage the crowdsourced user reviews to understand permission usage of apps.

AUTOREB [12] is the first work evaluating the risks of apps from user reviews. In AUTOREB, machine learning techniques were adopted to correlate user reviews with four security-related behaviors (e.g., spamming, financial issue, over-privileged permission, and data leakage). AUTOREB claimed that user reviews are valuable feedbacks from users by considering users' expectations. Considering that user reviews are collected in a crowdsourcing way, it can reflect users' expectations of apps better than app descriptions. Moreover, user reviews are kinds of real experiences of users

- *R. Wang is with the Key Laboratory of Aerospace Information Security and Trust Computing, School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China, and also with the School of Computer Science and Engineering, Nanyang Technological University, 639798, Singapore. E-mail: wangrun@whu.edu.cn.*
- *Z. Wang, B. Tang, L. Zhao, and L. Wang are with the Key Laboratory of Aerospace Information Security and Trust Computing, School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China. E-mail: {zbwang, tangbenxiao, leizhao, lnwang}@whu.edu.cn.*

Manuscript received 2 Feb. 2018; revised 7 May 2019; accepted 6 Aug. 2019. Date of publication 14 Aug. 2019; date of current version 3 Nov. 2020. (Corresponding authors: Zhibo Wang and Lina Wang.)
Digital Object Identifier no. 10.1109/TMC.2019.2934441

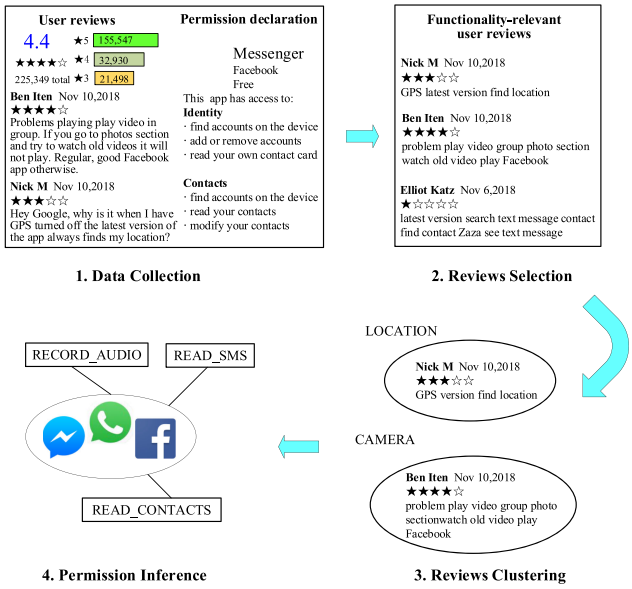

Fig. 1. The overview of SmartPI: understanding the permission implications from crowdsourced user reviews.

in enjoying apps' capabilities and contain many valuable information like descriptions of apps' functionalities. The app's functionalities information gives us an idea about the permissions that should be requested by the app [9]. Thus, we propose learning the permission implication from user reviews to evaluate the risks of using apps and try to explore why an app needs a permission. Due to the lack of professional knowledge, it is difficult for common users to infer the permission implications from user reviews. In this paper, we design a system, called SmartPI, to realize this objective automatically.

There are two key challenges in inferring permission implications from user reviews. The first challenge is how to identify the functionality-relevant user reviews from the noisy crowdsourced user reviews which are expressed for various purposes, such as bug reports, feature requests, and functionality descriptions. The permission documents in Android platform demonstrate that the application's functionality have relations with permission requests as AutoCog [9] claimed. Note that bug reports mainly tell us the application crash or exceptions occurred in use, but do not contain application's functionality descriptions. Although feature requests are mainly related to applications' functionalities, these functionalities don't exist in applications. Thus, only the functionality-relevant user reviews have relations with permission implications in the noisy user reviews.

The second challenge is how to accurately infer the permission implications from the functionality-relevant user reviews. The problem is challenging since different functionality-relevant user reviews may imply the same latent permissions. For example, *"I can find the nearby restaurants"*, *"I can book a wonderful hotel"* and *"a good navigator"* are all implying the permission of accessing location. Moreover, most of user reviews are short texts whose semantic is limited and sometimes are wrongly spelled.

In this paper, we design and implement a system, called SmartPI, to address the two challenges. The basic idea of SmartPI is to accurately infer the permission implications of apps from crowdsourced user reviews. As shown in Fig. 1, SmartPI consists of four phases: data collection, reviews selection, reviews clustering and permission inference. In the first phase, the metadata of user reviews and declared permissions of apps are collected from Google Play. In the second phase, representative words are extracted from Android platform documents and apps descriptions on Google Play to identify the functionality-relevant user reviews. In the third phase, the functionality-relevant user reviews are clustered based on their semantics with the topic model. Finally, a permission inference model is proposed to infer the permission implications of apps. We evaluate SmartPI with ten most dangerous permissions concerned by users, and conducts experiments on a real-world dataset with more than 20,000 popular applications, 4,247,769 reviews from 2,653,159 users on Google Play. Our results demonstrate that SmartPI can effectively identify permissions from user reviews.

Our main contributions are summarized as follows.

- We present the design of SmartPI by utilizing permission docs, API docs, and Apps' descriptions to effectively infer permission implications from crowdsourced user reviews.
- We extract representative words from permission docs, API docs, and Apps' descriptions by using NLP techniques to identify functionality-relevant user reviews.
- We present a permission inference model with an synthesis of user reviews' semantics and representative words for inferring the permission implication of the functionality-relevant user reviews.
- We evaluate the performance of SmartPI with a real-world dataset and the results demonstrate the effectiveness of SmartPI.
- SmartPI can evaluate the risk of an app with the collective wisdom and help users understand its permission requests.

The remainder of this paper is organized as follows. Section 2 discusses the related work. Section 3 presents the overview of our framework. Section 4 presents the design of SmartPI. Section 5 presents the implementation of SmartPI, and makes an evaluation of our framework. Finally, Section 6 concludes the paper.

2 Related Work

In this section, we briefly discuss the related work on malware detection, permission misuse in mobile security, mobile security researches using NLP techniques, and crowdsourcing in software engineering.

2.1 Malware Detection

Signature-based static analysis [4], [5] and behavior-based dynamic runtime analysis [6], [7] are two major techniques that are widely used in malicious application detection. FlowDroid [4] is a precise static taint analysis tool for Android application analysis, and it can reduce the false positives in dealing with application lifecycle or callback methods. Amandroid [5] is a general framework proposed to conduct static analysis for security vetting of Android apps, and it aims to address the problem of interactions among multiple components.

DroidScope [7] is a virtualization-based program analysis techniques. Both OS-level and Java-level semantics can be reconstructed by DroidScope. Taint analysis is also provided by DroidScope to collect detailed native and Dalvik instruction traces which can be used to analyze the data flow in app. TaintDroid [6] is the first work performing dynamic analysis in Android, and it presents a dynamic tainting component that enables the analyst to track sensitive data trace in execution. However, these techniques are all using program analysis techniques for malware detection and fail to consider app's functionalities in analysis. AppContext [13] proposed distinguishing whether an app's behavior is malicious or benign according to the contexts that trigger security-sensitive behaviors. To some extent, our work is similar to AppContext [13] which put the app into a specific context when analyze app's behaviors, but AppContext need to analyze the code of app with program analysis techniques. In this paper, we explore user expectations of apps, which can be used as a complement to program analysis in malicious behavior detection.

2.2 Permission Misuse in Mobile Security

In Android, permission is a particular security protection mechanism for protecting sensitive resources accessing. It is often used to remind users of the security or privacy behaviors that an app would carried. However, research [14] shows that less than 17 percent participants take care of permission requests reminding during app installing and only 3 percent participants have a correctly comprehension to some specific permissions. A lot of work focus on permission misuse in Android [15], [16], [17], [18], [19], because the permission misuse could bring some severe security and privacy threats to users. Geneiatakis et al. [17] propose a risk assessment framework to identify whether an app follows the least-privilege principle using static analysis and runtime information. Permlyzer [19] constructs a general-purpose framework using runtime analysis and static inspection to analyze the use of permission requests in app accurately. Stowaway [15] uses an automated testing techniques to detect over-privilege applications in Android.

Studies show that about one third applications are over-privileged, and these apps request uneeded permissions to access users' privacy data [11]. To address the problem of permission misuse in Android apps, a novel framework [20] is proposed by splitting existing permissions into massive of fine-grained permissions. A security framework [3] is designed and implemented to defeat collusion attacks which are caused by application-level privilege escalation attacks and the root cause is permission misuse.

Different from previous research work which use program analysis techniques to analyze the code of apps or modify the operating system, we employ NLP techniques without any modification of system or program analysis of apps.

2.3 Mobile Security Using NLP

NLP techniques are widely used in the security area. Recently, a lot of researchers use NLP techniques to solve mobile security problems, especially in Android platform. Whyper [8] aims to identify the sentence in app description which describe permission. However, Whyper only use the Android platforms' API documents and synonyms of words to achieve their goal. To overcome the limited semantic information of Whyper, AutoCog [9] leverages the corpus like Wikipedia to create a large-scale semantics database and applies natural language processing techniques to assess the description-to-permission fidelity in Android apps. Unlike previous work, TAPVerifier [10] takes privacy policy, bytecode, description, and permissions for analysis to revisit description-to-behavior fidelity. Different from Whyper and AutoCog, CHABADA [11] uses topic model to cluster apps' descriptions for identifying unexpected behaviors. Some researches like SUPOR [21] and UIPicker [22] use NLP techniques to identify sensitive user inputs of Android apps. In these work, text-based features are extracted from apps to infer the purpose of a permission request in apps using text analysis techniques [23]. AsDroid [24] identifies the contradiction between UI text and program behavior to detect stealthy behavior in apps. In AsDroid, NLP technique is used to infer the semantic of UI textual. Gao et al. [25] propose to detect emerging topics (e.g., new bugs) of mobile apps based on user reviews analysis.

In our work, we apply NLP techniques for review to permission inference. SmartPI is a complement to description-to-permission fidelity research like Whyper, AutoCog, and CHABADA, etc. These work are all aim to evaluate the risk of apps without using any program analysis techniques.

2.4 Crowdsourcing in Software Engineering

Crowdsourcing is widely used in software engineering for addressing various types of problems such as testing, software requirement analysis, feature request and so on. AUTOREB [12] proposes learning the relation between user review and security-related behaviors in a crowdsourcing way, but large amounts of manually labeled reviews are required in their work. User reviews are classified into bug reports, feature requests, user experiences, and ratings on app store using probabilistic techniques [26]. Cen et al. [27] propose a crowdsourcing ranking approach to evaluate the risks of applications from user reviews. Similar to these work, we also analyze user reviews which are collected in a crowdsourcing way for evaluating the risks of using apps.

3 Framework of SmartPI

In this section, we give a high-level overview of SmartPI and brifely describe its key components.

Fig. 2 shows the framework of SmartPI which trains a permission inference model in the offline training phase by considering user reviews, permission docs and API docs, and apps' descriptions, and predicts the permission implications of new user reviews in the online phase. We take an example user review which is collected from Google Play to illustrate how SmartPI works. The user review is written by Ben Iten who is a user of the application Facebook. The example user review is as follows (see the first phase data collection in Fig. 1).

"Problems playing video in groups. If you go to photos section and try to watch old videos it will not play. Regular, good Facebook app otherwise."

Android platform docs such as permission docs and API docs tell us the relation between needed permissions and App's functionalities. Thus, we extract permission-representative

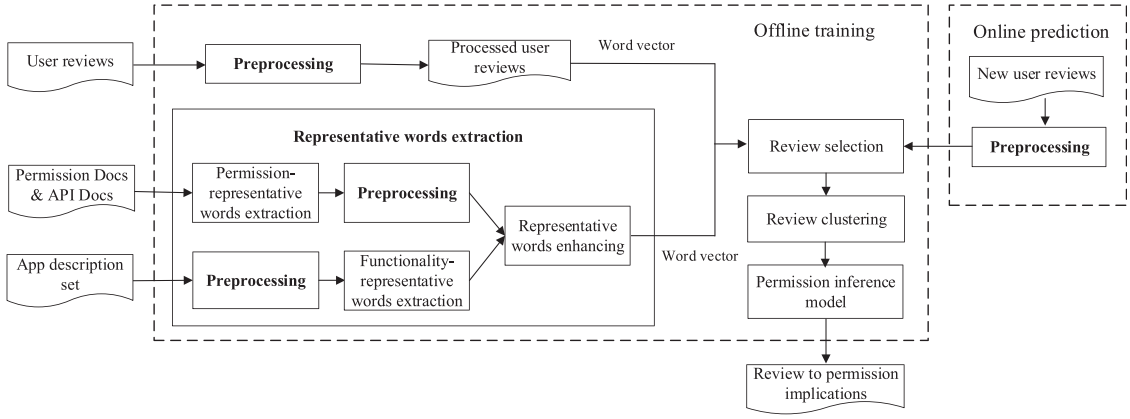

Fig. 2. The framework of SmartPI.

words of each permission from platform docs to identify functionality-relevant user reviews which imply permission use. It is not enough to extract permission-representative words from platform docs merely, so functionality-representative words are extracted from Apps' descriptions, mainly describe app's functionalities, for expanding permission-representative words. We mainly describe several key components of SmartPI.

Preprocessing. In order to accurately characterized the permission inference model, SmartPI takes user reviews as well as other sources into consideration, such as permission docs, API docs and Apps' descriptions. All these sources need to be preprocessed to remove noisy data for further analysis. In particular, the processing component includes language identification, stop words removal, and lemmatization, which selects user reviews written in English only, removes meaningless words in texts and normalized the words in texts. In our example user review, the user review is written in English, so the user review is preprocessed by stop words removal and lemmatization. The preprocessed example user review is a set of words as follows (see the second phase reviews selection in Fig. 1).

"*problem play video group photo section watch old video play Facebook*"

Representative Words Extraction. This component aims to build permission-representative word vector of each permission for functionality-relevant user reviews identification. The permission-representative words are extracted from platform docs such as permission docs and API docs. The number of permission-representative words is constrainted by the platform docs. Thus, two ways are adopted in SmartPI for expanding the permission-representative words. First, the synonyms of each word in permission-representative words are added. Second, functionality-representative words which are extracted from a large apps' description set on Google Play are also used to expand the permission-representative words.

Review Selection. This component aims to select the functionality-relevant user reviews which reflect permission implications from all users reviews based on whether the similarity distance between representative word vectors and user reviews' word vector is no less than a fixed threshold. The example user review is selected as functionality-relevant user review with the extracted representative-words (see Fig. 1).

Review Clustering. This component aims to cluster functionality-relevant user reviews for inferring the permission implication of user reviews. We adopt the topic model to discover the functionality semantics of user reviews in review clustering. The example user review describe the application Facebook providing the functionalities related to video and photo. In the review clustering analysis, the example user review belongs to the topic of camera based on its semantics.

Permission Inference Model. This component aims to infer the permission implication of user reviews in cluster after review clustering with a synthesis analysis of user reviews' semantics and permission-representative words. In the permission inference model, each cluster is correspond to a specific permission by taking the permission-representative words as new documents and determine the confidence of new documents that belong to topics. The topic camera represents the permission CAMERA in our permission inference model. Thus, the example user review which belongs to the topic camera implies the application need to request permission CAMERA in the Manifest file.

In offline training of SmartPI, the permission inference model is built for inferring the permission implication of the whole user reviews. In online prediction of SmartPI, when a new user review is submitted to app store, SmartPI estimates whether the review is functionality-relevant using our review selection component and determines which topic the identified functionality-relevant user review belongs to by our trained topic model in review clustering component. Finally, the trained permission inference model infers the new published user review to permission implication.

4 SmartPI Design

Fig. 3 shows the NLP analysis techniques used in SmartPI. In SmartPI, we adopt some NLP analysis techniques such as

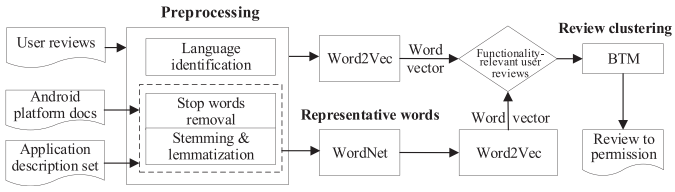

Fig. 3. NLP analysis techniques used in SmartPI.

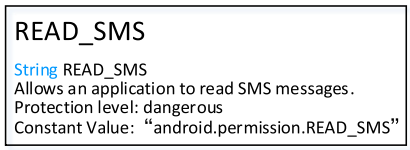

Fig. 4. Description of the permission *READ_SMS*.

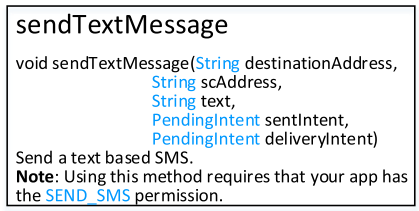

Fig. 5. Description of the sensitive API *sendTextMessage* corresponding to the permission *READ_SMS*.

words lemmatization, word embedding, topic model and so on. User reviews, Android platform docs and apps descriptions are first preprocessed by some basic NLP tasks like language identification, stop words removal and lemmatization for the normalization of words in texts. Word2Vec, a word embedding technique, is used to form the user reviews and representative words into word vectors for identifying functionality-relevant user reviews. The identified functionality-relevant user reviews are clustered by topic model, BTM, for building the permission inference model to infer the implicit relation between functionality-relevant user review and permission.

In this following, we present the design of each component and NLP analysis techniques used in detail.

4.1 Preprocessing

SmartPI takes users reviews as well as other sources into consideration, such as permission docs, API docs and Apps' descriptions. These texts must be preprocessed first for further analysis. In particular, we design three phases for the preprocessing component: language identification, stop words removal, and stemming and lemmatization.

Language Identification. The crowdsourced user reviews are written by people around the world with different languages and only the user reviews written in English are used in SmartPI. We use Google's Compact Language Detector [28] to realize this goal.

Stop Words Removal. Some words (e.g., I, the, of, my, it, to, from, at) are common in user reviews, permission docs, API docs and Apps' descriptions, but they have no specific meanings in sentences. Specifically, it may affect the results in review selection and review clustering if we keep them. Therefore, we build a dictionary of stop words based on *RANKS NL* [29] to help us remove these meaningless words from texts directly. In particular, there are more than one thousand stop words in our dictionary.

Stemming and Lemmatization. In English, a word is always in various forms in sentences under different contexts and scenarios, but they all originate from a basic form. For example, words like *drives*, *driving*, the base form of them is *drive*. The basic techniques, stemming and lemmatization, in text analysis is used to get the basic form of words for word normalization for further analysis. The NLTK library in Python is adopted in SmartPI to accomplish the phase of stemming and lemmatization.

4.2 Representative Words Extraction

Representative words are extracted from many data sources for identifying functionality-relevant user reviews which reflect permission implications. In SmartPI, the representative words are extracted from platform docs such as permission docs and API docs called permission-representative words. However, the number of permission-representative words is constrainted by platform docs. We propose expanding the permission-representative words in two ways.

One way is adding the synonyms of each words in permission-representative words. Here, we leverage the lexical database of English like WordNet [30] to obtain the synonym. In fact, some words are co-occurrence words of permission-representative words when describing apps' functionalities which reflect permission implications, but they are missed in permission-representative words extraction which consider the platform docs merely. To address this issue, we propose extracting representative words from Apps' descriptions, called functionality-representative words, to discover these words which are missed in permission-representative words extraction. Thus, the other way is expanding the permission-representative words with functionality-representative words.

4.2.1 Permission-Representative Words Extraction

We propose extracting permission-representative words from platform docs such as permission docs and API docs. Figs. 4 and 5 are examples of a permission and an API, respectively. The permission doc (see Fig. 4) is a programmers' guide in application's development and tells us the relation between apps' functionality and permission requests in application. The API doc (see Fig. 5) tells the developers the relation between permission and API. Thus, permission-representative words extracted from platform docs can be used to identify user reviews which describe apps' functionalities which reflect permission implications.

We parse the grammatical structure of sentences in permission documentation and API (corresponding to permissions [31]) documentation for permission-representative words extraction. In SmartPI, the grammatical structure of sentences is parsed by employing Stanford Parser [32], [33]. Fig. 6 shows the grammatical structure of the example sentence, *"Allows an application to read SMS messages"*, parsed by Stanford Parser. The Stanford Parser divides the sentence into several parts and the relation of each part is shown in typed dependencies (see. Fig. 6). The parse tree reveals each

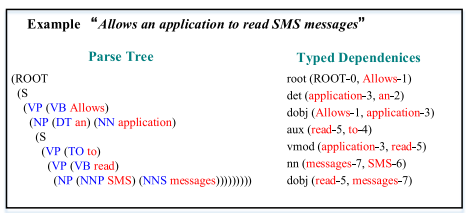

Fig. 6. Example of sentence parsing by Stanford Parser.

word with a *part-of-speech* (POS) tag such as verb, noun, adjective, pronoun, etc. In Fig. 6, *NN* means nouns, *NP* means noun phrase, *NNP* denotes proper noun, *NNS* indicates plural noun, and *VB* represents verb. After parsing by the Stanford Parser, the grammatical structure of the sentences in platform docs are obtained.

Following rules are adopted for permission-representative words extraction. First, words which are tagged as noun or noun phrase in parse tree are extracted directly to represent as permission-representative words. In SmartPI, we only consider the noun phrases, the number of nouns in them is less than 3. Second, word pair $\langle verb, noun\rangle$, parsed in typed dependency analysis, is also extracted as permission-representative words when the noun is the direct object of the verb. In the above example of Fig. 6, the permission-representative words of permission READ_SMS represents as $\{SMS, messages, application, \langle read, SMS\rangle, \langle read, messages\rangle\}$.

After words extraction, the permission-representative words should be processed by the text preprocessing component. Considering permission-representative words, the text preprocessing includes stop words removal and lemmatization. The platform docs in Android is written in English, so the phrase of language identification is ignored in text preprocessing. Here, *stop word* also indicates the common words that cannot provide essence meanings to permission implications. For example, words like "*application*", "*to*", and "*an*" in Fig. 6 are stop words that should be removed. *Stemming* and *Lemmatization* aim to find out the base form of each word. For example, the base form of word "*messages*" in Fig. 6 is "*message*". Thus, the permission-representative words of the above example in Fig. 6 is represented as $\{SMS, message, \langle read, SMS\rangle, \langle read, message\rangle\}$ after text preprocessing.

We use $\mathrm{R} = \{a_1, a_2, \ldots, a_\psi\}$ to represent the permission-representative words which are extracted from the permission docs, where ψ is the number of words in R. Let $\mathrm{I} = \{e_1, e_2, \ldots, e_\omega\}$ denote the permission-representative words which are extracted from the corresponding sensitive API docs, where ω is the number of words in I. Finally, the permission-representative words of permission h is represented as follows:

$$\mathrm{J}_h = \left\{ \mathrm{R}_h; \bigcup_{i=1}^{\chi} \mathrm{I}_i \right\}. \tag{1}$$

In Eq. (1), where $i = 1, 2, \ldots, \chi$, χ is the number of sensitive APIs corresponding to permission h.

4.2.2 Functionality-Representative Words Extraction

Some common words like "*region*" indicate the permission implication of ACCESS_COARSE_LOCATION or ACCESS_FINE_LOCATION implicitly, but these words cannot be found in permission-representative words which are extracted from Android platform docs merely. We found that these words are co-occurrence words in sentences which describe apps' functionalities and reflect permission implications, but they are not synonyms. Based on the above findings, we extract functionality-representative words from app description sentences which mainly describe apps' functionalities for expanding our permission-representative words. In SmartPI, we adopt an approach, integrating TF-IDF factors with TextRank [34], proposed by Tu et al. [35] to extract functionality-representative words from large app description set on Google Play.

In information retrieval and text mining, TF-IDF is often used to measure how important a word is to a document in a corpus. But the latent relation between candidate representative words in corpus is missing. To overcome the disadvantage of TF-IDF in representative words discovery, Tu et al. proposed an approach integrating TF-IDF with TextRank [35]. TextRank is a graph-based ranking model originated from PageRank model and widely used in many NLP tasks like keyword extraction, but it ignores the importance of words in document. We adopt their approach, taking the advantage of TF-IDF and TextRank, for functionality-representative words extraction from apps' description sentences.

Different from permission-representative words extraction, apps' descriptions should be processed by the text preprocessing phrase before functionality-representative words extraction. The text preprocessing includes language identification, stop words removal, and lemmatization. In functionality-representative words extraction, nouns and noun-phrases are first extracted from apps' description sentences using the Stanford Parser. We define $F = \{d_1, d_2, \ldots, d_\varsigma\}$ to represent the app description sentence after text preprocessing, where ς is the number of words in sentence and each element is a separate word which is noun or noun phrase. The TF-IDF value of each word d in app description sentence F is calculated by the following formula:

$$W_{tf\_idf}(F, d) = \frac{c(d)}{c(F)} * \lg\left(\frac{N(F)}{N(d)+1}\right), \tag{2}$$

where d is a word in app description sentence F, $c(d)$ is the frequency of d in F, $c(F)$ is the total number of words in F, $N(F)$ is the number of app description sentences in corpus L which is collected from Google Play, and $N(d)$ is the number of app description sentences in L containing d.

When consider TextRank, let $G = (V, E)$ represents an undirected graph, where V is the set of vertices and E is the set of edges. Let $V = \{w_1, w_2, \ldots, w_n\}$, where the element w is a noun or noun phrase and n is the total number of nouns and noun phrases in cropus. Let $E = \{(w_i, w_j)\}, i, j = 1, 2, \ldots, n$, where term (w_i, w_j) means word w_i and w_j co-occur in a app description sentence. The score of vertex in TextRank is defined as follows [34]:

$$S(L, w_i) = (1-\gamma)\frac{1}{|V|} + \gamma \sum_{(w_i, w_j)\epsilon E} \frac{c_t(w_i, w_j)}{O(w_j)} S(L, w_j). \tag{3}$$

In Eq. (3), $\gamma\epsilon(0, 1)$ is the damping factor, $O(w_j)$ represents the number of vertices that w_j points to other vertices, $c_t(w_i, w_j)$ indicates the number of term (w_i, w_j) co-occurrence in app description sentences in corpus L. To ensure that any graph structure can be coveraged, the probability $1-\gamma$ is granted to every vertices.

For $\forall w_i \in V$, the probability that w_i points to other vertices is $1-\gamma$. There is an unusual case that w_i has equal probabilities of pointing to other vertices in interactive process. It is reasonable that w_i should point to the vertices of higher correlation rather than lower correlation. Thus, we optimize the TextRank model with TF-IDF factor and it is defined as follows (see. Eq. (6)). The bigger TF-IDF factor means higher

correlation. δ is the TF-IDF factor in Eq. (4). In Eq. (5), $|L|$ means the number of description sentences in L.

$$\delta = 1 + W_{tf\_idf}(L, w_i) \tag{4}$$

$$W_{tf\_idf}(L, w_i) = \frac{1}{|L|}\sum_{j=1}^{|L|} W_{tf\_idf}(F_j, w_i) \tag{5}$$

$$S(L, w_i) = \delta(1-\gamma)\frac{1}{|V|} + \gamma \sum_{(w_i,w_j)\epsilon E} \frac{c_t(w_i, w_j)}{O(w_j)} S(L, w_j). \tag{6}$$

In graph construction, the window of the TextRank model is set to each app description sentences which are the basic processing and analysis unit. It means that the relation between these two words is built when the word pair co-occur in the same window. The arguments in training model are set as follows. The score of each vertex is set to an initial value of 1, γ is set to 0.85, the iterations is usually between 20 and 30, and the covergence threshold is set to 0.0001. Finally, top representative words are output based on the value of $S(L, w)$ until the model converges. In our work, we select the top 15 percent representative words ranked by $S(L, w)$ as functionality-representative words which are defined as Q.

4.2.3 Representative Words Enhancing

Due to the limitation of permission-representative words extraction (see. Section 4.2.1) which is obtained from platform docs merely. We propose enhancing the permission-representative words by adding synonyms and the co-occurrence words of functionality-representative words (see. Section 4.2.2).

The permission-representative words enhancing includes two ways. One way is using external English dictionary WordNet [30] to acquire the synonyms of each word in permission-representative words. For example, the synonyms of word *photo* are *picture* and *image* in WordNet. We add the synonyms to permission-representative words. The other way is using functionality-representative words for permission-representative words enhancing. For each word in permission-representative words, when their co-occurrence words, searching from the apps' description sentences corpus L, appeared in functionality-representative words and the co-occurrence times is no less than a fixed threshold ξ, they are used to enhance the permission-representative words. For instance, word *camera* and *photo* are co-occurred in the app description sentences corpus L. The word *camera* is a functionality-representative word, thus the word *camera* is used to expand the permission-representative word *photo*.

Let (v, v') represents a word pair, where v is a permission-representative word and v' is the co-occurrence word of v in app description sentences corpus L. Let $ratio(v, v')$ represents the frequency of word pair (v, v') co-occurred in corpus L. If the value of $ratio(v, v')$ is no lower than the threshold ξ, word v' will be added to permission-representative words for enhancing. In this paper, we set ξ to 0.03 by empirically finding the best values for it. $ratio(v, v')$ is calculated by the following formula:

$$ratio(v, v') = \frac{Cal(v, v')}{\sum_{i,j=0}^{\tau} Cal(v_i, v_j)}, \tag{7}$$

where $Cal(v, v')$ represents the times of word pair (v, v') co-occurred in L, τ is the total number of words in L.

4.3 Review Selection

In this section, we propose the review selection component to identify functionality-relevant user reviews which reflect permission implications from all user reviews based on the permission-representative words. Besides that, we also adopt sentiment analysis techniques to filter user reviews which request new features of apps.

User reviews are first processed by the preprocessing component and formed as feature vector using Word2Vec [36]. The enhanced permission-representative words are also formed as feature vector using Word2Vec. Let $\mathrm{U} = \{r_1, r_2, \ldots, r_\varpi\}$ denotes a user review after preprocessing, where ϖ is the number of words in U. Let J'_h represents the permission-representative words of permission h after enhancing. The similarity distance between U and J'_h after feature vectorization is calculated as follows:

$$sim(\mathrm{U}, \mathrm{J}'_\mathrm{h}) = \frac{\sum_{k=1}^{l} r'_k p'_k}{\sqrt{\sum_{k=1}^{l} {r'_k}^2}\sqrt{\sum_{k=1}^{l} {p'_k}^2}}. \tag{8}$$

In Eq. (8), the similarity between U and J'_h is measured using cosine distance, where l represents the length of normalized feature vector using Word2Vec, r'_k and p'_k represents the value of U and J'_h after vectorization, respectively.

In permission-representative words extraction, the permission-representative words of each permission are independent from each other. Thus, we need to iteratively calculate the similarity distance between user review after feature vectorization and these permission-representative words after feature vectorization and select the maximum one. When the maximum value is no lower than the threshold ϑ, the user review will be supposed as functionality-relevant one which reflects permission implications. The maximum distance is calculated by the following formula:

$$val(U, J') = \max_{h=1,2,\ldots,v} (sim(\mathrm{U}, \mathrm{J}'_h)), \tag{9}$$

where h represents the hth permission, v represents the number of permissions in permission-representative words extraction (see. Section 4.2.1), J' represents the permission-representative words set of all permissions after enhancing, and J'_h represents the permission-representative words of the hth permission after enhancing.

The identified functionality-relevant user reviews also contain feature requests reviews. These feature requests user reviews describe the non-existed functionalities of apps and should be filtered in permissions understanding.

A feature request user review always includes negative sentiment because the user is missing a feature of an app. Additionally, the future tense of verbs in user reviews can be used to indicate whether the review requests a missing feature of apps. We combine sentiment analysis with tense detection for filtering the feature request user reviews from the identified functionality-relevant reviews.

SentiStrength [37] assigns positive and negative user reviews with different sentiment score and is good at short text sentiment analysis. SmartPI uses SentiStrength to obtain the sentiment score for identifying the negative sentiment user reviews. The tense is extracted with part-of-sppech tagging which is provided in Stanford POS Tagger. In SmartPI, a functionality-relevant user review is regarded as feature

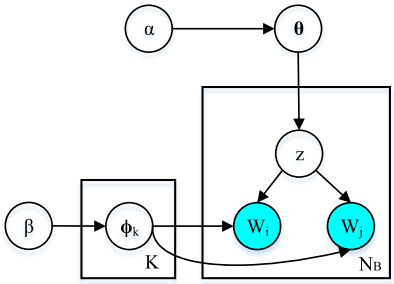

Fig. 7. Graphical representation of BTM [39].

request review when it includes negative sentiment and has more than 50 percent future tense verbs.

4.4 Review Clustering

In SmartPI, we cluster functionality-relevant user reviews for inferring permission implications. The implicit relation between user reviews and permissions is uncovered by clustering user reviews with the topic model.

Topic model is proposed to discover the latent semantic of texts and widely used in many fields such as recommender system, bioinformatics, etc. However, conventional topic models like Probabilistic Latent Semantic Analysis (PLSA) and Latent Dirichlet Allocation (LDA) cannot deal with short texts clustering well as their content is sparse. User reviews are usually short texts, and hence it is challenging for semantic analysis in clustering.

In this paper, we employ Biterm Topic Model (BTM) [38] for reviews clustering. BTM is a kind of topic model and experimental results show that it outperformed than PLSA and LDA in short texts clustering. In BTM, topics are learned over short texts by directly modeling the generation of all the biterms (e.g., word co-occurrence patterns) in the whole corpus [38]. Fig. 7 presents the graphical representation of BTM, where biterms are generated in a collection rather than documents like conventional topic models [38], [39].

In Fig. 7, w_i and w_j represents a biterm, which denotes an disorder co-occurrence word pair in a short text. N_B is the number of biterms in corpus. $B = \{b_1, b_2, \ldots, b_i, \ldots, b_m\}$ is the biterm set, where $b_i = (t_{i,1}, t_{i,2})$. The parameter $z \in [1, K]$ is a topic assignment variable, where K is the total number of topics in corpus. Let $D = \{u_1, u_2, \ldots, u_i, \ldots, u_n\}$ represents the corpus which is collected from Google Play, where u_i denotes the ith functionality-relevant user review. θ represents the K topics distribution in corpus, where $\theta = \{\theta_k\}_{k=1}^{K}$ with $\theta_k = P(z = k)$ and $\sum_{k=1}^{K} \theta_k = 1$. Let Φ represent the $K \times W$ matrix of the word distribution for topics and the kth row in Φ is denoted as ϕ_k. The hyperparameters α and β are dirichlet priors for θ and ϕ_k like the conventional topic model LDA.

The generative process of BTM (see Fig. 7) in SmartPI is described as follows. For each topic $k \in [1, K]$, BTM draws a word distribution of topic $\phi_k \sim Dir(\beta)$. A topic distribution $\theta \sim Dir(\alpha)$ is drawn for the whole corpus. For each biterm $b_i = (t_{i,1}, t_{i,2})$, BTM draws a topic $z \sim Multi(\theta)$ and two words biterm $t_{i,1}, t_{i,2} \sim Multi(\phi_z)$. Finally, the joint probability of a biterm b_i is calculated by the following formula [38] with a given multinomial distribution θ and Φ:

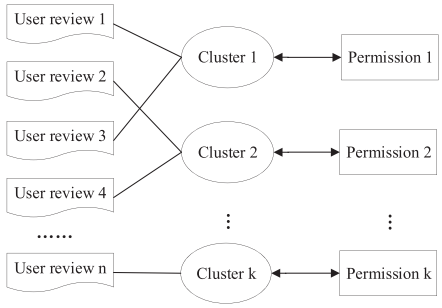

Fig. 8. The inference procedure of review to permission.

$$\begin{aligned} P(b_i|\theta, \Phi) &= \sum_{k=1}^{K} P(t_{i,1}, t_{i,2}, z_i = k|\theta, \Phi) \\ &= \sum_{k=1}^{K} P(z_i = k|\theta_k) P(t_{i,1}, t_{i,2}\,|z_i = k, \phi_{k,t_{i,1},t_{i,2}}) \\ &= \sum_{k=1}^{K} \theta_k \phi_{k,t_{i,1}} \phi_{k,t_{i,2}}. \end{aligned} \tag{10}$$

Here, the user review can be regraded as document like in conventional topic model. Thus, the topic distribution of a user review u is calculated as follows:

$$P(z|u) = \sum_{j=1}^{N_u} P(z, b_j|u) = \sum_{j=1}^{N_u} P(z|b_j, u) P(b_j|u). \tag{11}$$

In Eq. (11), N_u represents the number of biterms in user review u, b_j represents the biterms in u, $j \epsilon [1, N_u]$. In BTM, Gibbs Sampling is used to infer the multinomial parameters of $\{\phi, \theta\}$. Please refer to the published literature [38] for the details of BTM.

In our model training, the number of topics is set to the number of permissions (see. Section 5.1). The dirichlet priors α and β are set to 5 and 0.01 in BTM.

4.5 Permission Inference Model

The permission implication of functionality-relevant user reviews in each cluster is inferred by our proposed permission inference model with an synthesis analysis of user reviews' semantics and permission-representative words of permissions. In Fig. 8, the procedure of review to permission inference, a mapping between cluster and permission is built by determining the confidence of permission-representative words that belong to topics (topic model of review clustering in Section 4.4).

In our permission inference model, we predict the confidence of permission that belongs to topics by taking the permission-representative words as new documents. Let J'_h represents the permission-representative words of the hth permission after enhancing. We calculate the value of $P(z = k|J'_h)$ using the fixed parameters $\{\phi, \theta\}$ after model training in Section 4.4. The topic that J'_h belongs is calculated by the following formula:

$$\begin{aligned} k &= \underset{h=1,2,\ldots,t}{arg\,max}\, P(z = k|J'_h) \\ &= \underset{h=1,2,\ldots,t}{arg\,max} \sum_{j=1}^{N_y} P(z = k|b_j) P(b_j|J'_h). \end{aligned} \tag{12}$$

TABLE 1
The Statistical Details of Dataset T (Raw Data Collected from Google Play) and T' (Functionality-Relevant User Reviews after Review Selection)

Dataset	#app	#review	#author	mean	max	min
T	20,941	4,247,769	2,653,159	202	800	1
T'	19,014	361,276	312,315	19	259	1

The mean, max, and min are statistical data of the number of reviews per app.

In Eq. (12), N_h represents the number of biterms in permission-representative words of permission h, ι represents the number of permissions. The function $argmax$ is used in Eq. (12) to match permission to topic by considering that the characteristics of the ten evaluated permissions are significantly different and each permission maps to their own particular topic. Finally, the permission h belongs to topic k.

5 Implementation and Performance Evaluation

In this section, we present the implementation and experimental results of SmartPI, and compare its performance with existing system. We evaluate its performance in two aspects: functionality-relevant user reviews identification and review to permission implication inference. We attempt to answer the following two questions.

- *Q1:* What is the accuracy of the functionality-relevant user review identification?
- *Q2:* Can we trust the permission inferred by SmartPI?

5.1 Data Collection and Permission Selection

All the experimental data (e.g., app descriptions and user reviews) are collected from Google Play. Application ID is a unique identification for app on Google Play. In our work, we collected the app IDs from *PlayDrone* [40], a scalable crawler for Google Play. Our experimental dataset T includes more than 20,000 applications and 4 million user reviews. Table 1 shows the statistical data of dataset T; the largest number of user reviews for an app is 800, which is the maximum value setting in our crawler; the average value is 202. More than 63.3 percent apps' user reviews are between 100 and 400 in dataset T.

We also collected app descriptions which is used to extract functionality-representative word for review selection. The app description corpus L contains more than 1.4 million records over 49 different app categories. We trust that the corpus L can provide enough content for us to expand the permission-representative words for functionality-relevant user reviews selection.

In Android, there are more than 24 permissions defined as dangerous level in official developers documents, but not all of them are worried by common users. In SmartPI, we select ten security/privacy-related permissions for evaluation, which are shown in Table 2. In permission selection, we need to answer a question, *which permissions are concerned by most common users and may cause security threats or privacy leakage to mobile devices*. To answer this question, we take a survey in our campus and the survey results show that "*SMS*", "*CONTACTS*", "*LOCATION*", "*AUDIO*", and "*CAMERA*" are the keypoints attracting users' attention. Recently, some researchers find that malware developers can guess the password of application accessing or the password of lock screen by collecting sensor data in smartphone such as gyroscope sensor, acceleration sensor [41]. The dangerous permission BODY_SENSORS used to collect the sensitive data about the heath of users' body, such as heart rate. Thus, most of the common users cared about whether the app collects sensor data.

TABLE 2
Ten Permissions Evaluated in Our Experiments and the Corresponding Percentage of Apps Requesting the Permission

Permission	#App (Percentage %)
SEND_SMS	4.68%
READ_SMS	1.21%
CALL_PHONE	10.16%
ACCESS_COARSE_LOCATION	23.95%
ACCESS_FINE_LOCATION	26.16%
READ_CONTACTS	6.78%
RECORD_AUDIO	7.82%
BODY_SENSORS	1.2%
CAMERA	15.24%
READ_PHONE_STATE	35.71%

TABLE 3
Statistical Data of Dataset S and S' in Review Selection Evaluation

Dataset	#app	#Review	#Label(Y)	#Label(N)
S	1,000	18,732	2,807	15,925
S'	100	1,602	224	1,378

The columns "Label(Y)" and "Label (N)" mean that the review is functionality-relevant or not.

We evaluate SmartPI with ten permissions by considering the survey results and our understanding of the security and privacy threats to users. Table 2 shows us the ten permissions. We also find that more than 52.36 percent apps have requested at least one of these permissions on Google Play.

5.2 Q1: What is the Accuracy of the Functionality-Relevant User Review Selection

Experiment Setting. To evaluate the functionality-relevant review selection, the validation dataset S and the testing dataset S' are obtained from dataset T. As for the ground truth, we ask some students with strong background of security in our lab to label them functionality-relevant or not. The statistical data of dataset S and S' are shown in Table 3.

In SmartPI, we set the threshold $\vartheta = 0.6$ in review selection component (see Section 4.3) after several iterative validation tests between validation dataset S and testing dataset S'. In our test validation experiment for finding the best threshold, the relation between threshold setting and precision of functionality-relevant user reviews selection shows in Fig. 9.

Evaluation Measurement. In review selection, we evaluate our results with four essential factors namely precision, recall, F-score, and accuracy. Typically, we take true positives, false positives, false negative, and true negatives as TP, FP, FN, and TN. TP means that our system correctly identify a functionality-relevant review; FP means that the review without any functionality is wrongly identified as a functionality-relevant review; FN means that our system can correctly identify a

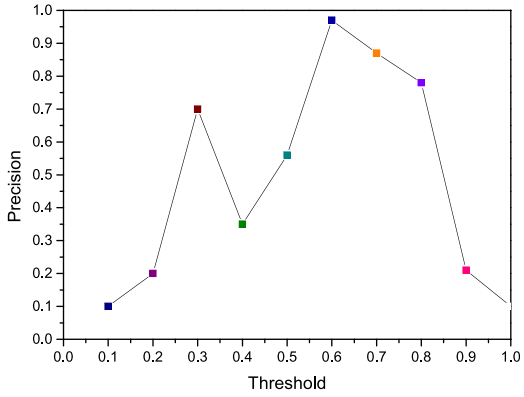

Fig. 9. The relationship between precision and threshold ϑ in functionality-relevant user reviews selection.

non-functionality-relevant review; TN means that our system wrongly identify a non-functionality-relevant review as a functionality-relevant review. These values are used to measure the performance of our approach in review selection. In particular, we have

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F - score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Experimental Results. The enhanced word vector is formed as feature for identifying functionality-relevant user reviews by combining the permission-representative words and functionality-representative words. The samples of permission-representative words after enhancing are shown in Table 4. Let's take permission *camera* for example, the representative word *video* and *sharing* cannot be extracted from platform docs like permission docs and API docs as permission-representative words, but these two words imply the app's functionality of video recording and social media sharing like image or video sharing. Thus, these two words are representative words for functionality-relevant user reviews selection. In our work, we carefully trace the source of these two words and found that they are from a piece of app descriptions in Google Play. Result shows that we adopted to extract the functionality-representative words for expanding the permission-representative words can solve the limited semantics problem of permission-representative words and significantly increase the accuracy of review selection.

Our experimental result shows that SmartPI can efficiently identify functionality-relevant user reviews with an average precision, recall, F-score and accuracy of 83, 82, 82, and 81 percent, respectively. We also use the permission-representative words without enhancing to identify the functionality-relevant user reviews and find that the recall is smaller than 50 percent. The low recall means that a lot of functionality-relevant user review will be missed if we use the permission-representative words before enhancing as feature for review selection. Obviously, our proposed approach of permission-representative words enhancing is valuable in review selection.

5.3 Q2: Credibility of Review to Permission Inference

Experiment Setting. The permission implication of user review is inferred with the permission inference model after user reviews clustering by topic model. In our permission inference model, each topic represents the corresponding permission (see Table 5). In Table 5, the second column denotes the assigned topic name based on the semantics of topics and the third column lists the top three representative words of each topic.

In topic model training, the arguments of α and β in BTM are set to 5 and 0.01. The number of topics, is set to 10 which is the number of permissions in our experiment evaluation (see Section 5.1). The training data of our topic model, BTM, is functionality-relevant user reviews which are identified from dataset T and formed as dataset T' (see Table 1). Let $P(z|u)$ denotes the probability propotion of review u under topic z. We assume the review u belong to topic z when the value of $P(z|u)$ is no lower than a threshold φ. We set φ to 0.4 by empirically finding the best value of it.

We compare SmartPI with an existing systems, Autocog [9], which also inferred the permission requested in app from app descriptions. In particular, we randomly select 200 popular apps from Google Play to evaluate the results of understanding the permission requested in apps from user reviews and app descriptions separately.

Experimental Results. Table 6 shows the result of review to permission inference, we find that SmartPI achieves a good

TABLE 4
The Samples of Enhanced Permission-Representative Words

Permission	Representative words sample
SEND_SMS	$SMS, message, text, sm, contact, book, communication, \langle send, message \rangle$
READ_SMS	$SMS, message, text, sm, contact, book, communication, \langle read, message \rangle$
CALL_PHONE	$call, address, answer, conference, telephone, ring, number, emergency$
ACCESS_COARSE_LOCATION	$approximate\ location, map, gps, estimate\ location, route, restaurant$
ACCESS_FINE_LOCATION	$precise\ location, accurate\ location, bank, travel, park, area, downtown$
READ_CONTACTS	$\langle read, contacts \rangle, message, number, friend, group, backup, member, call$
RECORD_AUDIO	$\langle record, audio \rangle, recorder, playback, voice, speech, talk, report, stereo$
BODY_SENSORS	$temperature, gyroscope, compass, game, light, sensor, speed, accelerator$
CAMERA	$photographer, camera, video, montage, photo, image, sharing, webcam, paint$
READ_PHONE_STATE	$phone\ number, ongoing\ call, bluetooth\ cellular\ network\ information$

TABLE 5
The Resulting Topics with the Corresponding Permission

ID	Assigned Topic Name	Top 3 Representative Words of Topics	Permission
0	send messages	SMS, message, send	SEND_SMS
1	read messages	SMS, message, read	READ_SMS
2	telephone call	call, telephone, phone	CALL_PHONE
3	access approxi-mate location	location, gps, navigation	ACCESS_COARSE_LOCATION
4	access precise location	location, map, gps	ACCESS_FINE_LOCATION
5	read contacts	contacts, friend, message	READ_CONTACTS
6	record audio	audio, speech, voice	RECORD_AUDIO
7	sensor	sensor, heart rate, comp-ass	BODY_SENSORS
8	camera	photo, image, picture	CAMERA
9	device state in-formation	cellular network, phone number, phone call	READ_PHONE_STATE

TABLE 6
Results of Review to Permission Inference

System	Permission	TP	FP	FN	TN	Prec(%)	Rec(%)	F(%)	Accu(%)
SmartPI	SEND_SMS	51	6	6	83	89.5%	89.5%	89.5%	91.8%
	READ_SMS	48	4	5	82	92.3%	90.6%	91.4%	93.5%
	CALL_PHONE	46	3	6	95	93.9%	88.5%	91.1%	94%
	ACCESS_COARSE_LOCATION	38	6	8	110	86.4%	82.6%	84.4%	91.4%
	ACCESS_FINE_LOCATION	56	7	3	92	88.9%	94.9%	91.8%	93.7%
	READ_CONTACTS	58	7	9	75	89.2%	86.6%	87.9%	89.3%
	RECORD_AUDIO	89	7	11	83	89.2%	86.6%	87.9%	89.3%
	BODY_SENSORS	119	9	8	67	93%	93.7%	93.3%	91.6%
	CAMERA	63	6	7	78	91.3%	90%	90.6%	91.6%
	READ_PHONE_STATE	69	2	3	88	97.2%	95.8%	96.5%	96.9%
	Total	637	57	66	853	91.8%	90.6%	91.2%	92.4%
AutoCog	SEND_SMS	51	7	4	76	87.9%	92.7%	90.3%	92%
	READ_SMS	56	6	7	86	90%	88.9%	89.6%	91.6%
	CALL_PHONE	21	3	8	34	87.5%	72.4%	79.2%	83.3%
	ACCESS_COARSE_LOCATION	36	5	9	87	87.8%	80%	83.7%	89.8%
	ACCESS_FINE_LOCATION	48	5	7	93	90.6%	87.3%	88.9%	92.2%
	READ_CONTACTS	69	8	6	81	89.6%	92%	90.8%	91.5%
	RECORD_AUDIO	107	8	6	83	93%	94.7%	93.9%	93.1%
	BODY_SENSORS	67	7	15	60	90.5%	81.7%	85.9%	85.2%
	CAMERA	62	8	6	78	88.6%	91.2%	89.6%	90.9%
	READ_PHONE_STATE	63	9	6	79	87.5%	91.3%	89.4%	90.4%
	Total	580	66	74	757	89.8%	88.7%	89.2%	90.5%

performance about the precision, recall, F-score, and accuracy in permission BODY_SENSORS, READ_PHONE_STATE, and CALL_PHONE inference than pervious work like AutoCog, the state-of-art work in permission implication inference of app description sentences. The experimental results show that SmartPI performs an average precision, recall, F-socre, and accuracy as 91.8, 90.6, 91.2, and 92.4 percent in inferring 10 permissions from user reviews. However, AutoCog achieves the average precision, recall, F-score and accuracy as 89.8, 88.7, 89.2, and 90.5 percent.

This is because SmartPI extracts the representative words from the domain knowledge database like permission docs, API docs, and Apps' descriptions. The domain knowledge database is good at identifying the functionality-relevant user reviews which imply the permission implications. However, AutoCog leverages the general corpus Wikipedia to create a semantic database. The general corpus database is less effective than the domain knowledge database when identifying the functionality-relevant user reviews. In addition, app descriptions, written by apps' developers, are more formally than user reviews which is written by numerous users. The user reviews' implicit semantics is hardly discovered by the Explicit Semantic Analysis (ESA) [42] which is adopted in AutoCog.

We randomly select 200 popular apps from Google Play to compare the claimed app description and user reviews for confirming which one can better reflect the pemission usage of apps. The experimental results show that SmartPI can identify 97 percent claimed permissions from user reviews. However, the apps' description based approach AutoCog can only identify 61 percent claimed permissions from apps' descriptions. Therefore, the permission usage of apps can be better reflected by user reviews than the claimed description of apps.

We also manually analyze apps' descriptions of these selected 200 apps and find that apps' descriptions are mainly tell users the basic functionalities, the new features and the unique features of apps. The apps' description is provided by developers merely and their writing style and contents are different. User reviews are collected in a crowdsourcing way and they are the real experiences of common users in using apps. Thus, user reviews of popular apps are more valuable than

apps' descriptions in understanding the permission requests of apps.

6 Conclusion

In this paper, we present SmartPI by leveraging the power of the crowdsourced user reviews to understand why an app requests permissions. We extract features from platform docs and large app descriptions with NLP techniques to identify functionality-relevant user reviews which reflect permission implications of apps. Topic model is adopted for inferring the permission implication from functionality-relevant user reviews. We evaluate our approach on real-dataset which is collected from GooglePlay. Experimental results demonstrate that crowdsourced user reviews can better reflect the permission requested by apps than app description in previous work.

As for new apps, SmartPI can infer the permission implications of apps from application descriptions. In SmartPI, the application description can be treated as a special user review provided by developers. SmartPI can identify whether the new application requests unneeded permission by comparing the declared permissions in manifest files with the inferred permissions of apps which provide similar functionalities from user reviews.

To benign and malicious user reviews, SmartPI can effectively identify the permission implications of them. SmartPI suggests a potential risky app when some permissions can be inferred from malicious user reviews but failed in analyzing app descriptions which can be regarded as a special user review written by developers. Additionally, SmartPI is a complement to previous work AUTOREB in identifying the malicious user reviews which indicate permission implications.

SmartPI assumes that users are honest and the submitted user reviews are trusted. Malicous attackers could fool SmartPI by crafting spam user reviews with the information of permission implication. We will consider how to address this problem in our future work.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grants No. 61876134, No. U1836112, No. 61872274, and No. 61672394), the National Key Research and Development Program of China (No. 2016YFB0801100), and the Fundamental Research Funds for the Central Universities (Grants No. 2042019gf0098, and No. 2042018gf0043).

References

[1] "GSMA Mobile Economy 2017," Jul. 2017. [Online]. Available: https://www.gsma.com/mobileeconomy/global/2017
[2] J. H. Saltzer, "Protection and the control of information sharing in Multics," *Commun. ACM*, vol. 17, no. 7, pp. 388–402, 1974.
[3] S. Bugiel, L. Davi, A. Dmitrienko, T. Fischer, A.-R. Sadeghi, and B. Shastry, "Towards taming privilege-escalation attacks on android," in *Proc. NDSS*, vol. 17, 2012, Art. no. 19.
[4] S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. Le Traon, D. Octeau, and P. McDaniel, "Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps," *ACM SIGPLAN Notices*, vol. 49, no. 6, pp. 259–269, 2014.
[5] F. Wei, S. Roy, X. Ou, et al., "Amandroid: A precise and general inter-component data flow analysis framework for security vetting of android apps," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2014, pp. 1329–1341.
[6] W. Enck, P. Gilbert, S. Han, V. Tendulkar, 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," *ACM Trans. Comput. Syst.*, vol. 32, no. 2, 2014, Art. no. 5.
[7] L.-K. Yan and H. Yin, "DroidScope: Seamlessly reconstructing the OS and Dalvik semantic views for dynamic Android Malware analysis," in *Proc. USENIX Secur. Symp.*, 2012, pp. 569–584.
[8] R. Pandita, X. Xiao, W. Yang, W. Enck, and T. Xie, "WHYPER: Towards automating risk assessment of mobile applications," in *Proc. 22nd USENIX Conf. Secur.*, 2013, pp. 527–542.
[9] 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. Secur.*, 2014, pp. 1354–1365.
[10] L. Yu, X. Luo, C. Qian, and S. Wang, "Revisiting the description-to-behavior fidelity in android applications," in *Proc. IEEE 23rd Int. Conf. Softw. Anal. Evol. Reengineering*, vol. 1, pp. 415–426, 2016.
[11] A. Gorla, I. Tavecchia, F. Gross, and A. Zeller, "Checking app behavior against app descriptions," in *Proc. 36th Int. Conf. Softw. Eng.*, 2014, pp. 1025–1035.
[12] D. Kong, L. Cen, and H. Jin, "AUTOREB: Automatically understanding the review-to-behavior fidelity in android applications," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 530–541.
[13] W. Yang, X. Xiao, B. Andow, S. Li, T. Xie, and W. Enck, "Appcontext: Differentiating malicious and benign mobile app behaviors using context," in *Proc. IEEE/ACM 37th IEEE Int. Conf. Softw. Eng.*, vol. 1, pp. 303–313, 2015.
[14] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner, "Android permissions: User attention, comprehension, and behavior," in *Proc. 8th Symp. Usable Privacy Secur.*, 2012, Art. no. 3.
[15] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, "Android permissions demystified," in *Proc. 18th ACM Conf. Comput. Commun. Secur.*, 2011, pp. 627–638.
[16] X. Wei, L. Gomez, I. Neamtiu, and M. Faloutsos, "Permission evolution in the android ecosystem," in *Proc. 28th Annu. Comput. Secur. Appl. Conf.*, 2012, pp. 31–40.
[17] D. Geneiatakis, I. N. Fovino, I. Kounelis, and P. Stirparo, "A Permission verification approach for android mobile applications," *Comput. Secur.*, vol. 49, pp. 192–205, 2015.
[18] Z. Fang, W. Han, and Y. Li, "Permission based android security: Issues and countermeasures," *Comput. Secur.*, vol. 43, pp. 205–218, 2014.
[19] W. Xu, F. Zhang, and S. Zhu, "Permlyzer: Analyzing permission usage in android applications," in *Proc. IEEE 24th Int. Symp. Softw. Rel. Eng.*, 2013, pp. 400–410.
[20] J. Jeon, K. K. Micinski, J. A. Vaughan, A. Fogel, N. Reddy, J. S. Foster, and T. Millstein, "Dr. Android and Mr. Hide: Fine-grained permissions in android applications," in *Proc. 2nd ACM Workshop Secur. Privacy Smartphones Mobile Devices*, 2012, pp. 3–14.
[21] J. Huang, Z. Li, X. Xiao, Z. Wu, K. Lu, X. Zhang, and G. Jiang, "SUPOR: Precise and Scalable Sensitive User Input Detection for Android Apps," in *Proc. 24th USENIX Secur. Symp.*, 2015, pp. 977–992.
[22] Y. Nan, M. Yang, Z. Yang, S. Zhou, G. Gu, and X. Wang, "UIPicker: User-input privacy identification in mobile applications," in *Proc. 24th USENIX Conf. Secur. Symp.*, 2015, pp. 993–1008.
[23] H. Wang, J. Hong, and Y. Guo, "Using text mining to infer the purpose of permission use in mobile apps," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, 2015, pp. 1107–1118.
[24] J. Huang, X. Zhang, L. Tan, P. Wang, and B. Liang, "Asdroid: Detecting stealthy behaviors in android applications by user interface and program behavior contradiction," in *Proc. 36th Int. Conf. Softw. Eng.*, 2014, pp. 1036–1046.
[25] C. Gao, J. Zeng, M. R. Lyu, and I. King, "Online app review analysis for identifying emerging issues," in *Proc. IEEE/ACM 40th Int. Conf. Softw. Eng.*, 2018, pp. 48–58.
[26] W. Maalej and H. Nabil, "Bug report, feature request, or simply praise? on automatically classifying app reviews," in *Proc. IEEE 23rd Int. Requirements Eng. Conf.*, 2015, pp. 116–125.
[27] L. Cen, D. Kong, H. Jin, and L. Si, "Mobile app security risk assessment: A crowdsourcing ranking approach from user comments," in *Proc. SIAM Int. Conf. Data Mining*, 2015, pp. 658–666.
[28] "Compact Language Detector," Jul. 2017. [Online]. Available: http://code.google.com/p/chromium-compact-language-detector
[29] "Ranks nl," Jul. 2017. [Online]. Available: http://www.ranks.nl/stopwords

[30] G. A. Miller, "WordNet: A lexical database for English," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.
[31] K. W. Y. Au, Y. F. Zhou, Z. Huang, and D. Lie, "Pscout: Analyzing the android permission specification," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2012, pp. 217–228.
[32] M.-C. De Marneffe, B. MacCartney, C. D. Manning, et al., "Generating typed dependency parses from phrase structure parses," in *Proc. 5th Int. Conf. Lang. Res. Eval.*, vol. 6, no. 2006, pp. 449–454, 2006.
[33] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer, "Feature-rich part-of-speech tagging with a cyclic dependency network," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics Human Lang. Technol. - Vol. 1*, 2003, pp. 173–180.
[34] R. Mihalcea and P. Tarau, "TextRank: Bringing order into texts," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Jul. 25–26, 2004, pp. 404–411.
[35] T. Shouzhong and H. Minlie, "Mining microblog user interests based on TextRank with TF-IDF factor," *J. China Universities Posts Telecommun.*, vol. 23, no. 5, pp. 40–46, 2016.
[36] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.
[37] M. Thelwall, K. Buckley, and G. Paltoglou, "Sentiment strength detection for the social web," *J. Amer. Soc. Inf. Sci. Technol.*, vol. 63, no. 1, pp. 163–173, 2012.
[38] X. Yan, J. Guo, Y. Lan, and X. Cheng, "A biterm topic model for short texts," in *Proc. 22nd Int. Conf. World Wide Web*, 2013, pp. 1445–1456.
[39] X. Cheng, X. Yan, Y. Lan, and J. Guo, "BTM: Topic modeling over short texts," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 12, pp. 2928–2941, Dec. 2014.
[40] N. Viennot, E. Garcia, and J. Nieh, "A measurement study of google play," in *Proc. ACM SIGMETRICS Perform. Eval. Rev.*, vol. 42, no. 1, pp. 221–233, 2014.
[41] Z. Xu, K. Bai, and S. Zhu, "Taplogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors," in *Proc. 5th ACM Conf. Secur. Privacy Wireless Mobile Netw.*, 2012, pp. 113–124.
[42] E. Gabrilovich and S. Markovitch, "Computing semantic relatedness using wikipedia-based explicit semantic analysis," in *Proc. 20th Int. Joint Conf. Artif. Intell.*, vol. 7, pp. 1606–1611, 2007.

Run Wang received the BM degree in management science and engineering from the Anhui University of Technology, China, in 2013, and the ME and PhD degrees both in computer science from Wuhan University, China, in 2015 and 2018, respectively. He has been working as a postdoctoral research fellow with Nanyang Technological University, Singapore, since April 2019. His research interests include mobile security and privacy, and AI security.

Zhibo Wang received the BE degree in automation from Zhejiang University, China, in 2007, and the PhD degree in electrical engineering and computer science from the University of Tennessee, Knoxville, in 2014. He is currently a professor with the School of Cyber Science and Engineering, Wuhan University, China. His currently research interests include internet of things, network security, and privacy protection. He is a senior member of the IEEE and a member of the ACM.

Benxiao Tang received the BE degree in software from Wuhan University, China, in 2013. He is currently working toward the PhD degree in information security in the School of Cyber Science and Engineering, Wuhan University. His research interests include mobile privacy and program analysis. He is a student member of the IEEE.

Lei Zhao received the BE and PhD degrees both in computer science from Wuhan University, China, in 2007 and 2012, respectively. He worked as an assistant professor with Wuhan University from 2013 to 2015. He is currently an associate professor with the School of Cyber Science and Engineering, Wuhan University, China. His currently research interests include software security, software analysis, and system protection. He is a member of the IEEE.

Lina Wang received the MS and PhD degrees from Northeastern University, in 1989 and 2001, respectively. She is currently a professor and PhD supervisor with the School of Cyber Science and Engineering, Wuhan University. She is a member of the China Computer Federation. She has authored 50 research papers and founded 12 authorized patents. Her current research interests include system security and steganalysis. She is a member of the IEEE.