AutoCog: Measuring the Description-to-permission Fidelity in Android Applications

Zhengyang Qu[1], Vaibhav Rastogi[1], Xinyi Zhang[2], Yan Chen[1],
Tiantian Zhu[3], Zhong Chen[4]
[1]Department of Electrical Engineering and Computer Science,Northwestern University, Illinois, USA
[2]Software School, Fudan University, Shanghai, China
[3]Software College, Northeastern University, Shenyang, China
[4]Wind Mobile,Toronto, Ontario, Canada
{zhengyangqu2017, vrastogi}@u.northwestern.edu, ychen@northwestern.edu,
10302010070@fudan.edu.cn, laozhutt@gmail.com, zhong.chen@utoronto.ca

ABSTRACT

The booming popularity of smartphones is partly a result of application markets where users can easily download wide range of third-party applications. However, due to the open nature of markets, especially on Android, there have been several privacy and security concerns with these applications. On Google Play, as with most other markets, users have direct access to natural-language descriptions of those applications, which give an intuitive idea of the functionality including the security-related information of those applications. Google Play also provides the permissions requested by applications to access security and privacy-sensitive APIs on the devices. Users may use such a list to evaluate the risks of using these applications. To best assist the end users, the descriptions should reflect the need for permissions, which we term *description-to-permission fidelity*. In this paper, we present a system AutoCog to automatically assess description-to-permission fidelity of applications. AutoCog employs state-of-the-art techniques in natural language processing and our own learning-based algorithm to relate description with permissions. In our evaluation, AutoCog outperforms other related work on both performance of detection and ability of generalization over various permissions by a large extent. On an evaluation of eleven permissions, we achieve an average precision of 92.6% and an average recall of 92.0%. Our large-scale measurements over 45,811 applications demonstrate the severity of the problem of low description-to-permission fidelity. AutoCog helps bridge the long-lasting usability gap between security techniques and average users.

Categories and Subject Descriptors

D.0 [**Software**]: General

CCS'14, November 3–7, 2014, Scottsdale, Arizona, USA.

http://dx.doi.org/10.1145/2660267.2660287.

Keywords

Android, Natural language processing, Machine learning, Mobile, Google play, Permissions

1. INTRODUCTION

Modern operating systems such as Android have promoted global ecosystems centered around large repositories or marketplaces of applications. Success of these platforms may in part be attributed to these marketplaces. Besides serving applications themselves, these marketplaces also host application metadata, such as descriptions, screenshots, ratings, reviews, and, in case of Android, permissions requested by the application, to assist users in making an informed decision before installing and using the applications. From the security perspective, applications may access users' private information and perform security-sensitive operations on the devices. With the application developers having no obvious trust relationships with the user, these metadata may help the users evaluate the risks in running these applications.

It is however generally known [14] that few users are discreet enough or have the professional knowledge to understand the security implications that may be derived from metadata. On Google Play, users are shown both the application descriptions and the permissions[1] declared by applications. An application's description describes the functionality of an application and should give an idea about the permissions that would be requested by that application. We call this *description-to-permission fidelity*. For example, an application that describes itself as a social networking application will likely need permissions related to device's address book. A number of malware and privacy-invasive applications have been known to declare more permissions than their purported functionality warrants [10, 33].

With this belief that descriptions and permissions should generally correspond, we present AutoCog, a system that automatically identifies if the permissions declared by an application are consistent with its description. AutoCog has multi-fold uses.

- Application developers can use this tool to receive an early, automatic feedback on the quality of descriptions

[1]In Android, security-sensitive system APIs are guarded by permissions, which applications have to declare and which have to be approved at install-time.

so that they improve the descriptions to better reflect the security-related aspects of the applications.
- End users may use this system to understand if an application is over-privileged and risky to use.
- Application markets can deploy this tool to bolster their overall trustworthiness.

The key challenge is to gather enough semantics from descriptions in natural language to reason about the permissions declared. We apply state-of-the-art techniques from natural language processing (NLP) for sentence structure analysis and computing semantic relatedness of natural language texts. We further develop our own learning-based algorithm to automatically derive a model that can be queried against with descriptions to get the expected permissions.

AutoCog is a substantial advancement over the previous state-of-the-art technique by Pandita et al. [26], who have also attempted to develop solutions with the same goals. Their tool called Whyper is primarily limited by the use of a fixed vocabulary derived from the platforms' API documents and the English synonyms of keywords there. Our investigations show that Whyper's methodology is inherently limited regarding the following issues: (*a*) *Limited semantic information*: not all textual patterns associated with a permission can be extracted from API documents, e.g., <"*find*", "*branch atm*"> relate to location permissions and <"*scan*", "*barcode*"> relate to the permission for accessing the camera in our models but cannot conceivably be found from API documents; (*b*) *Lack of associated APIs*: certain permissions do not have associated APIs so that this methodology cannot be used; and (*c*) *Lack of automation*: it is not clear how the techniques could be automated. We have confirmed these limitations with Whyper's authors as well.

Our methodology is radically different from Whyper's as is evident from the following contributions of this paper.

- *Relating descriptions and permissions.* We design a novel learning-based algorithm for modeling the relatedness of descriptions to permissions. Our algorithm correlates textual semantic entities (second contribution) to the declared permissions. It is noteworthy that the model is trained entirely from application descriptions and declared permissions over a large set of applications without depending on external data such as API documents, so that we do not have the problems of limited semantic information or lack of associated APIs from the very outset. Both training and classification are completely automatic.
- *Extracting semantics from descriptions.* We utilize state-of-the-art NLP techniques to automatically extract semantic information from descriptions. The key component for semantics-extraction in our design is Explicit Semantic Analysis (ESA), which leverages big corpuses like Wikipedia to create a large-scale semantics database, and which has been shown to be superior to dictionary-based synonyms and other methods [16] and is being increasingly adopted by numerous research and commercial endeavors. Such superior analysis further largely mitigates the problem of limited semantic information.
- *System prototype.* We design and implement an end-to-end tool called AutoCog to automatically extract relevant semantics from Android application descriptions and permissions to produce permission models. These models are used to measure description-to-permission fidelity: given an application description, a permission model outputs whether the permission is expected to be declared by that application. If the answer is yes, AutoCog further provides relevant parts of description that warrant the permission. This tool is published on Google Play[2] and the backend data is available on our web portal[3].

We further have the following evaluation and measurement highlights.

- *Evaluation.* Our evaluation on a set of 1,785 applications shows that AutoCog outperforms the previous work on detection performance and ability of generalization over various permissions by a large extent. AutoCog closely aligns with human readers in inferring the evaluated permissions from textual descriptions with an average precision of 92.6% and average recall of 92.0% as opposed to previous state-of-the-art precision and recall of 85.5% and 66.5% respectively.
- *Measurements.* Our findings on 45,811 applications using AutoCog show that the description-to-permissions fidelity is generally low on Google Play with only 9.1% of applications having permissions that can all be inferred from the descriptions. Moreover, we observe the negative correlation between fidelity and application popularity.

The remainder of this paper is organized as follows. Section 2 gives further motivation of our work and presents a brief background and problem statement. Next we cover AutoCog design in detail in Section 3, followed by the implementation aspects in Section 4. Section 5 deals with the evaluation of AutoCog and introduces our measurement results. We have relevant discussion and related work in Sections 6 and 7. Finally, we conclude our work in Section 8.

2. BACKGROUND AND PROBLEM STATEMENT

2.1 Background

Android is the most popular smartphone operating system with over 80% market share [1]. It introduces a sophisticated permission-based security model, whereby an application declares a list of permissions, which must be approved by the user at application installation. These permissions guard specific functionalities on the device, including some security and privacy-sensitive APIs such as access contacts.

Modern operating systems such as Android, iOS, and Windows 8 have brought about the advent of big, centralized application stores that host third-party applications for users to view and install. Google Play, the official application store for Android, hosts both free and paid applications together with a variety of metadata including the title and description, reviews, ratings, and so on. Additionally, it also provides the user with the ability to study the permissions requested by an application.

2.2 Problem Statement

The application descriptions on Google Play are a means for the developers to communicate the application functionality to the users. From the security and privacy standpoint, these descriptions should thus indicate the reasons for the

[2]`https://play.google.com/store/apps/details?id=com.version1.autocog`
[3]`http://python-autocog.rhcloud.com`

permissions requested by an application, either explicitly or implicitly[4]. We call it *fidelity* of descriptions to permissions.

As stated in Section 1, Android applications often have little in their descriptions to indicate to the users why they need the permissions declared. Specifically, there is frequently a gap between the access of the sensitive device APIs by the applications and their stated functionality. This may not always be out of malicious intent; however users are known to be concerned about the use of sensitive permissions [12]. Moreover, Felt et al. [14] show that few users are careful enough or able to understand the security implications derived from the metadata. In this work we thus look into the problem of *automatically assessing the fidelity of the application descriptions with respect to the permissions.*

Detection of malicious smartphone applications is possible through static/run-time analysis of binaries [9, 18, 32]. However, the techniques to evaluate whether application oversteps the user expectation are still lacking. Our tool can assist the users and other entities in the Android ecosystem assess whether the descriptions are faithful to the permissions requested. AutoCog may be used by users or developers individually or deployed at application markets such as Google Play. It may automatically alert the end users if an application requests more permissions than required for the stated functionalities. The tool can provide useful feedback about the shortcomings of the descriptions to the developers and further help bolster the overall trustworthiness of the mobile ecosystem by being deployed at the markets.

As for automatically measuring description-to-permission fidelity, we need to deal with two concepts: (a), the *description semantics*, which relates to the meaning of the description, and (b), the *permission semantics*, which relates to the functionality provided (or protected) by the permission. The challenges in solving our problem therefore lie in:

- *Inferring description semantics*: Same meaning may be conveyed in a vast diversity of natural language text. For example, the noun phrases "*contact list*", "*address book*", and "*friends*" share similar semantic meaning.
- *Correlating description semantics with permission semantics*: A number of functionalities described may map to the same permission. For example, the permission to access user location might be expressed with the texts "*enable navigation*", "*display map*", and "*find restaurant nearby*". The need for permission to write to external disk can be implied as "*save photo*" or "*download ringtone*".

In AutoCog, we consider the decision version of the problem stated above: *given a description and a permission, does the description warrant the declaration of the permission?* If AutoCog answers yes, it provides the sentences that warrant the permission, thus assisting users in reasoning about the requested permission. As a complete system, AutoCog solves this decision problem for each permission declared.

Whyper [26] is a previous work with goals similar to ours. Whyper correlates the description and permission semantics by extracting natural language keywords from an external source, Android API documents. Since APIs and permissions can be related together [2], the intuition is that keywords and patterns expressed in the API documentation will also be found in the application descriptions and are therefore adequate in representing the respective permissions. Based on our investigation, the methodology has the following fundamental limitations:

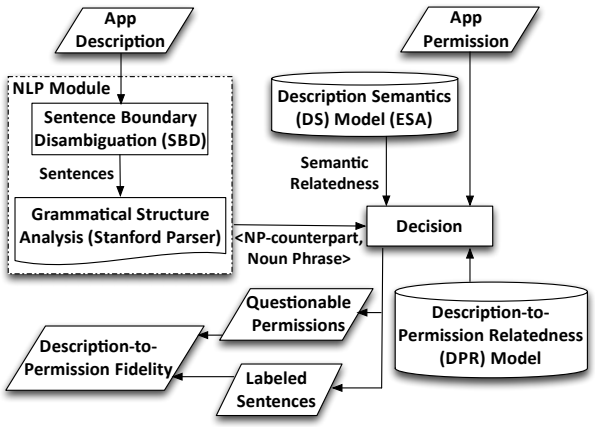

Figure 1: Overall architecture of AutoCog

- *Limited semantic information*: the API documents are limited in the functionality they describe and so Whyper cannot cover a complete set of semantic patterns correlated with permissions. For example, in our findings, the pattern <"*deposit*", "*check*"> is related to the permission CAMERA with high confidence but cannot be extracted from API documents. The mobile banking applications, such as Bank of America[5], support depositing by snapping its photo with the device's camera. Analysis on this issue in detail will be shown in Section 5.2.
- *Lack of associated APIs*: certain sensitive permissions such as RECEIVE_BOOT_COMPLETED do not have any associated APIs [2]. It is thus not possible to generate the correlated textual pattern set with the API documents.
- *Lack of automation*: Whyper's extraction of patterns from API documents involved manual selection to preserve the quality of patterns; what policies could be used to automate this process in a systematic manner is an open question.

Our learning-based approach automatically discovers a set of textual patterns correlated with permissions from the descriptions of a rich set of applications, hence enabling our description-to-permission relatedness model to achieve a complete coverage over the natural language texts with great diversity. Besides, the training process works directly on descriptions. So we easily overcome the limitations of the previous work as stated above.

3. SYSTEM DESIGN

Figure 1 gives an architectural overview of AutoCog. The description of the application is first processed by the NLP module, which disambiguates sentence boundaries and analyzes each sentence for grammatical structure. The output of the NLP module is then passed in together with the application permissions into the decision module, which, based on models of description semantics and description-to-permission relatedness outputs the questionable permissions that are not warranted from the description and the sentences from which the other permissions may be inferred.

[4]By implicit, we mean that the need for permission is evident from stated functionality.

[5]`https://play.google.com/store/apps/details?id=com.infonow.bofa`

These outputs together provide description–to-permission fidelity. This section provides a detailed design of each of the modules and the models that constitute AutoCog.

3.1 NLP Module

The goal of the NLP module is to identify specific constructs in the description such as noun and verb phrases and understand relationship among them. Use of such related constructs alleviates the shortcomings of simple keyword-based analysis. The NLP module consists of two components, sentence boundary disambiguation and grammatical structure analysis.

3.1.1 Sentence boundary disambiguation (SBD)

The whole description is split into sentences for subsequent sentence structure analysis [21, 30]. Characters such as ".", ":", "-", and some others like "*", "♠", "♢" that may start bullet points are treated as sentence separators. Regular expressions are used to annotate email addresses, URLs, IP addresses, Phone numbers, decimal numbers, abbreviations, and ellipses, which interfere with SBD as they contain the sentence separator characters.

3.1.2 Grammatical structure analysis

We leverage Stanford Parser [31] to identify the grammatical structure of sentences. While our design depends on constructs provided by the Stanford Parser, it is conceivable that other NLP parsers could be used as well.

We first use the Stanford Parser to output *typed dependencies*, which are semantic hierarchies of sentences, i.e., how different parts of sentences depend on each other. As illustrated in Figure 2, the dependencies are triplets: *name of the relation*, *governor* and *dependent*. *Part of Speech (PoS)* tagging additionally assigns a part-of-speech tag to each word; for example, a verb, a noun, or an adjective. The results are fed into *phrase parsing* provided by Stanford Parser to break sentences into phrases, which could be noun phrases, verb phrases or other kinds of phrases. We obtain a hierarchy of marked phrases and tagged words for each sentence.

The governor-dependent pair provides the knowledge of logic relationship between various parts of sentence, which provides the guideline of our ontology modeling. The concept of ontology is a description of things that exist and how they relate to each other. In our experience, we find the following ontologies, which are governor-dependent pairs based on noun phrase, to be most suitable for our purposes.

- Logical dependency between verb phrase and noun phrase potentially implies the actions of applications performing on the system resources. For example, the pairs <"scan", "barcode"> and <"record", "voice"> reveal the use of permissions camera and recording.
- Logical dependency between noun phrases is likely to show the functionalities mapped with permissions. For instance, users may interpret the pairs <"scanner", "barcode"> and <"note", "voice"> as using camera and microphone.
- Noun phrase with own relationship (possessive, such as "your", followed by resource names) is recognized as requesting permissions. For example, the CAMERA and RECORD_AUDIO permissions could be revealed by the pairs <"your", "camera"> and <"own", "voice">.

We extract all the noun phrases in the leaf nodes of the hierarchical tree output from grammatical structure analysis. For each noun phrase, we record all the verb phrases and noun phrases that are its ancestors or siblings of its ancestors. We also record the possessive, if the noun phrase itself contains the own relationship. For the sake of simplicity, we call the extracted verb phrases, noun phrases, and possessives as *np-counterpart* for the target noun phrase. The noun-phrase based governor-dependent pairs obtained signify direct or indirect dependency. The example hierarchy tree for sentence "*Search for a place near your location as well as on our interactive maps*" is shown in Figure 2 with the pairs extracted: <"search", "interactive map">, <"our", "interactive map">, <"search", "place">, <"search", "location">, <"place", "location">, and <"your", "location">.

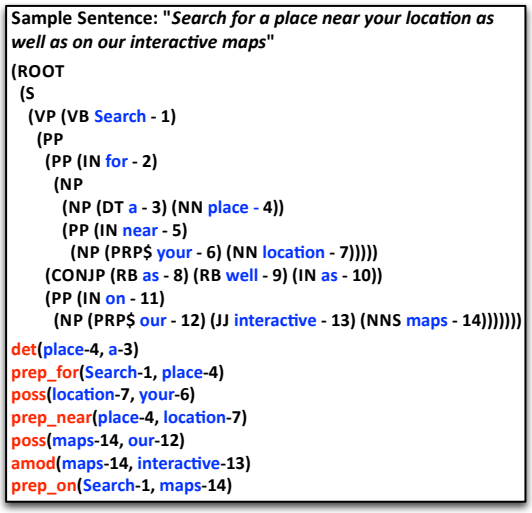

Figure 2: Example output of Stanford Parser

We process these pairs to remove stopwords and named entities. *Stopwords* are common words that cannot provide much semantic information in our context, *e.g.*, "the", "and", "which", and so on. Named entities include names of persons, places, companies, and so on. These also do not communicate security-relevant information in our context. To filter out named entities, we employ *named entity recognition*, a well-researched NLP topic, also implemented in Stanford Parser. The remaining words are normalized by lowercasing and *lemmatization* [8]. Example normalizations include "better" → "good" and "computers" → "computer".

3.2 Description Semantics (DS) Model

The goal here is to understand the meaning of a natural language description, i.e., how different words and phrases in a vocabulary relate to each other. Similarly meaning natural language descriptions can differ vastly; so such an analysis is necessary. Our model is constructed using *Explicit Semantic Analysis (ESA)*, the state of the art for computing semantic relatedness of texts [16]. The model is used directly by the decision module and also for training the description-to-permission relatedness model discussed in Section 3.3.

ESA is an algorithm to measure the semantic relatedness between two pieces of text. It leverages big document corpuses such as Wikipedia as its knowledge base and constructs a vector representation of text. In ESA, each (Wiki) article is called a concept, and transformed into a weighted vector of words within the article. As processing an input article, ESA computes the relatedness of the input to ev-

Table 1: Distribution of noun phrase patterns

Pattern	#Noun Phrase (Percentage %)
Noun	1,120,850 (52.37 %)
Noun + Noun	414,614(19.37 %)
Adjective + Noun	278,785 (13.03 %)
Total	1,814,249 (84.77 %)

Pattern of noun phrase; Number/percentage of noun phrases in the pattern within 2,140,225 noun phrases extracted from 37,845 applications

ery concept, i.e. projects the input article into the concept space, by the common words between them. In NLP and information retrieval applications, ESA computes the relatedness of two input articles using the cosine distance between the two projected vectors.

We choose ESA because it has been shown to outperform other known algorithms for computing semantic relatedness such as WordNet and latent semantic analysis [16]. We offer intuitive reasons of out-performance over WordNet as this has been used in WHYPER. First, WordNet-based methods are inherently limited to individual words, and adoption for comparing longer text requires an extra level of sophistication [24]. Second, considering words in context allows ESA to perform word sense disambiguation. Using WordNet cannot achieve disambiguation, since information about synsets (sets of synonyms) is limited to a few words; while in ESA, concepts are associated with huge amounts of text. Finally, even for individual words, ESA offers a much more detailed and quantitative representation of semantics. It maps the meaning of words/phrases to a weighted combination of concepts, while mapping a word in WordNet amounts to simple lookup, without any weight.

3.3 Description-to-Permission Relatedness (DPR) Model

Description-to-permission relatedness (DPR) model is a decisive factor in enhancing the accuracy of AUTOCOG. We design a learning-based algorithm by analyzing the descriptions and permissions of a large dataset of applications to measure how closely a noun-phrase based governor-dependent pair is related to a permission. The flowchart for building the DPR model is shown in Figure 3. We first leverage ESA to group the noun phrases with similar semantics. Next, for each permission, we produce a list of noun phrases whose occurrence in descriptions is positively related to the declaration of that permission. Such phrases may potentially reveal the need for the given permission. In the third stage, we further enhance the results by adding in the np-counterparts (of the noun-phrase based governor-dependent pairs) and keeping only the pairs whose occurrence statistically correlates with the declaration of the given permission.

3.3.1 Grouping Noun Phrases

A noun phrase contains a noun possibly together with adjectives, adverbs, etc. During the learning phase, since analyzing long phrases is not efficient, we consider phrases of only three patterns: single noun, two nouns, and noun following adjective (Table 1). In our dataset of 37,845 applications, these patterns account for 85% of the 302,739 distinct noun phrases. We further note that we focus on these restricted patterns only during DPR model construction; all noun phrases are considered in the decision module of AUTOCOG, which checks whether the description of application indicates a given permission. The DS model, which is also employed during decision-making, can match longer patterns with similarly meaning noun phrases grouped here. Hence the negative effect of such simplification is negligible.

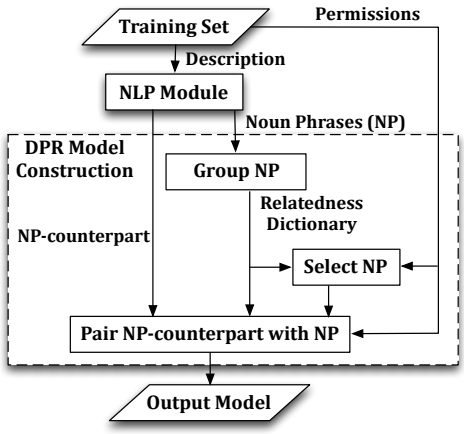

Figure 3: Flowchart of description-to-permission relatedness (DPR) model construction

We construct a *semantic relatedness score matrix* leveraging DS model with ESA. Each cell in the matrix depicts the semantic relatedness score between a pair of noun phrases. Define the *frequency* of noun phrase to be the number of applications whose descriptions contain the noun phrase. As constructing the semantic relatedness score matrix has quadratic runtime, it is not scalable and efficient. We filter out noun phrases with low frequencies from this matrix, as the small number of samples cannot provide enough confidence in our frequency-based measurement. If a low-frequency phrase is similar to a high-frequency phrase, our decision process will not be affected as the decision module employs DS model. We choose a threshold; only phrases with frequency above 15 are used to construct the matrix. The number of such phrases in our dataset is 9,428 (3.11%).

Using the semantic relatedness score matrix, we create a *relatedness dictionary*, which maps a given noun phrase to a list of noun phrases, all of which have a semantic relatedness score higher than the threshold θ_g. The interpretation is that the given noun phrase may be grouped with its list of noun phrases as far as semantics is concerned. Our implementation takes θ_g to be 0.67. The lists also record the corresponding semantic relatedness scores for later use. A sample dictionary entry of the noun phrase "*map*" is:
<"map", [("map", 1.00), ("map view", 0.96), ("interactive map", 0.89),...]>

3.3.2 Selecting Noun Phrases Correlated With Permissions

Whether a certain noun phrase is related to a permission is learnt statistically from our dataset. If a permission *perm* and a noun phrase *np* appear together (i.e., *perm* in permission declarations and *np* in the description) in a high number of applications, it implies a close relationship between the two. This is however not trivial; some noun phrases (*e.g.*, "game" and "application") may occur more frequently than others, biasing such calculations. Moreover, some noun phrases may actually be related to permissions but statisti-

cal techniques may not correlate them if they occur together in only a few cases in the dataset. The latter is partially resolved by leveraging the relatedness dictionary from the previous step. Based on existing data mining techniques [25], we design a quality evaluation method that (a) is not biased to frequently occurring noun phrases, and (b) takes into account semantic relatedness between noun phrases to improve the statics of meaningful noun phrases that occurs less than often. For the permission $perm$ and the noun phrase np, the variables in the learning algorithm are defined as:

MP(perm, np): An application declares $perm$. Either np or any noun phrase with the semantic relatedness score to np above the threshold θ_g is found in the description. This variable will increase by 1, if np is in the description, or it will increase by the maximal relatedness score of the noun phrase(s) related to np.

MMP(perm, np): An application does NOT declare $perm$. Either np or any noun phrase with the semantic relatedness score to np above the threshold θ_g is found in the description. This variable will increase by 1, if np is in the description, or it will increase by the maximal relatedness score of the noun phrase(s) related to np.

PR(perm, np): The ratio of $MP(perm, np)$ to the sum of $MP(perm, np)$ and $MMP(perm, np)$:

$$PR(perm, np) = \frac{MP(perm, np)}{MP(perm, np) + MMP(perm, np)}.$$

AVGPR(perm): The percentage of all the applications in our training set that request $perm$.

INCPR(perm, np): This variable measures the increment of the probability that $perm$ is requested with the presence of np or its related noun phrases given the unconditional probability as the baseline:

$$INCPR(perm, np) = \frac{PR(perm, np) - AVGPR(perm)}{AVGPR(perm)}.$$

MMNP(perm, np): An application declares $perm$. This variable will increase by 1, if none of np and noun phrases related to it in the Relatedness Dictionary are found in the description.

NPR(perm, np): The ratio of $MP(perm, np)$ to the sum of $MP(perm, np)$ and $MMNP(perm, np)$:

$$NPR(perm, np) = \frac{MP(perm, np)}{MP(perm, np) + MMNP(perm, np)}.$$

AVGNP(np): Expectation on the probability that one description contains np or related noun phrases over the training set. Assume the total number of applications is M. This variable is expressed as:

$$AVGNP(np) = \frac{\Sigma_{i=1}^{i=M} \lambda_i}{M},$$

where λ_i equals 1, if np is in the description of the *i-th* application. Or it equals to the maximal semantic relatedness score of its related noun phrase(s) found in description. If neither np nor noun phrases related to it in the Relatedness Dictionary are found, $\lambda_i = 0$.

INCNP(perm, np): This variable measures the growth on the probability that one description includes np or the related noun phrases with the declaration of $perm$ given expectation as the baseline:

$$INCNP(perm, np) = \frac{NPR(perm, np) - AVGNP(np)}{AVGNP(np)}.$$

Semantic relatedness score is taken as weight in the calculations of variables $MP(perm, np)$ and $MMP(perm, np)$, which groups the related noun phrases and resolves the minor case issue. We should note that $INCPR(perm, np)$ and $INCNP(perm, np)$ evaluate the quality of np by the growth of the probabilities that $perm$ is declared and np (or noun phrases related to np) is detected in description with the average level as baseline. This design largely mitigates the negative effect caused by the intrinsic frequency of noun phrase. To roundly evaluate the quality of np of describing $perm$, we define the $Q(perm, np)$, which is the harmonic mean of $INCPR(perm, np)$ and $INCNP(perm, np)$:

$$Q(perm, np) = \frac{2 \cdot INCPR(perm, np) \cdot INCNP(perm, np)}{INCPR(perm, np) + INCNP(perm, np)}.$$

np with negative values of $INCPR$ or $INCNP$ is discarded as it shows no relevance to $perm$. Each permission has a list of noun phrases, arranged in descending order by the quality value. The *top-k* noun phrases are selected for the permission. We set k=500 after checking the distribution of quality value for each permission. It is able to give a relatively complete semantic coverage of the permission. Increasing the threshold k excessively would enlarge the number of noun-phrase based governor-dependent pairs in the DPR model. So it would reduce the efficiency of AutoCog in matching the semantic meaning for the incoming descriptions.

3.3.3 Pair np-counterpart with Noun Phrase

By following the procedure presented in Section 3.3.2, we can find a list of noun phrases closely related to each permission. However, simply matching the permission with noun phrase alone fails to explore the context and semantic dependencies, which increases false positives. Although a noun phrase related to "*map*" is detected in the example sentence below, it does not reveal any location permission.

"*Retrieve Running Apps" permission is required because, if the user is not looking at the widget actively (for e.g. he might using another app like Google Maps)*"

To resolve this problem, we leverage Stanford Parser to get the knowledge of context and typed dependencies. For each selected noun phrase np, we denote as $G(np)$ the set of noun phrases that have semantic relatedness scores with np higher than θ_g. Given a sentence in description, our mechanism identifies any noun phrase $np' \in G(np)$ and records each np-counterpart nc (recall that np-counterpart was defined as a collective term for verb phrases, noun phrases, and possessives for the target noun phrase), which has direct/indirect relation with np'. For each noun-phrase based governor-dependent pair $< nc, np >$, let the total number of descriptions where the pair $< nc, np' >$ is detected be SP. In the SP applications, let the number of application requesting the permission is tc. We keep only those pairs for which (1) $tc/SP > Pre_T$, (2) $SP > Fre_T$, where Pre_T and Fre_T are configurable thresholds. Thus we maintain the precision and the number of samples large enough to yield statistical results with confidence.

3.4 Decision

In DPR model, each permission has a list of related pairs of np-counterpart nc_{dpr} and noun phrase np_{dpr}, which reveal the security features of the permission. For an input application whose description has to be checked, the NLP module extracts the pairs of np-counterpart nc_{new} and noun phrase

np_{new} in each sentence. We leverage the DS model to measure the semantic relatedness score $RelScore(txt_A, txt_B)$ between the two texts txt_A and txt_B. The sentence is identified as revealing the permission, if $< nc_{new}, np_{new} >$ is matched with a pair $< nc_{dpr}, np_{dpr} >$ by fulfilling the conduction:

$$RelScore(nc_{new}, nc_{dpr}) > \Upsilon,$$
$$RelScore(np_{new}, np_{dpr}) > \Theta.$$

Here, Υ and Θ are the thresholds of the semantic relatedness score for np-counterparts and noun phrases. The sentences indicating permissions will be annotated. Besides, AutoCog finds all the questionable permissions, which are not warranted in description.

4. IMPLEMENTATION

NLP Module: We use the NLTK library in Python and regular expression matching to implement the SBD. NLTK is also used for removing stopwords and normalizing words using lemmatization based on WordNet. Stanford Named Entity Recognizer is used for removing named entities.

DS and DPR Models: Noun phrases are classified by frequency. High-frequency noun phrases are grouped based on semantic relatedness score by utilizing the library `esalib`[6]. This library is the only currently maintained, open-source implementation of ESA that we could find. Our training algorithm on descriptions and permissions of large-scale applications selects the semantic patterns, which strongly correlate with the target permission by leveraging the frequency-based measurement and ESA. Our current implementation pairs np-counterpart of length one (noun, verb, and possessive) with noun phrases. The np-counterpart could be easily extended to multiple words, possibly with a few considerations about maximum phrase length, and so on.

Overall, We implement AutoCog with over 7,000 lines of code in Python and 500 lines of code in Java.

5. EVALUATION

We first describe our dataset and methodology for collecting sensitive permissions. Then, AutoCog's accuracy is evaluated by comparing with Whyper [26]. Finally, we discuss our measurements, which investigate the overall trustworthiness of market and the correlation between description-to-permission fidelity and application popularity.

5.1 Permission Selection and Dataset

The Android APIs have over a hundred permissions. However, some permissions such as the permission VIBRATE, which enables vibrating the device, may not be as sensitive as, for example, the permission RECORD_AUDIO, which enables accessing the microphone input. It is not so useful to identify permissions that are not considered sensitive. The question to ask then is, *what permissions are the users most concerned about from the security/privacy perspective?*

Felt et al. [12] surveyed 3,115 smartphone users about 99 risks and asked the participants to rate how upset they would be if a given risk occurred. We infer 36 Android platform permissions from the risks with highest user concerns. Since we focus here on third-party applications, we first remove from this list the Signature/System permissions, which are granted only to applications that are signed with the device manufacturer's certificate. Seven permissions were removed as a result. The 29 remaining permissions are arranged in descending order by the percentage of applications requesting it in our dataset, which is collected randomly. We select the top 14 permissions in our evaluation, because the ground-truth of our evaluation relies on readers to identify whether sentences in application description imply sensitive permissions; the consequent human efforts make it difficult to review large number of descriptions.

Table 2: Permissions used in evaluation

Permission	#App (Percentage %)
WRITE_EXTERNAL_STORAGE	30384 (80.29 %)
ACCESS_FINE_LOCATION	16239 (42.91 %)
ACCESS_COARSE_LOCATION	15987 (42.24 %)
GET_ACCOUNTS	12271 (32.42 %)
RECEIVE_BOOT_COMPLETED	9912 (26.19 %)
CAMERA	6537 (17.27 %)
GET_TASKS	6214 (16.42 %)
READ_CONTACTS	5185 (13.70 %)
RECORD_AUDIO	4202 (11.10 %)
CALL_PHONE	3130 (8.27 %)
WRITE_SETTINGS	3056 (8.07 %)
READ_CALL_LOG	2870 (7.58 %)
WRITE_CONTACTS	2176 (5.74 %)
READ_CALENDAR	817 (2.16 %)

Permission name; Number/percentage of applications request the permission within 37,845 applications;

We collected the declared permissions and descriptions of 37,845 Android applications from Google Play in August 2013 for the purpose of training the DPR model and evaluate AutoCog's accuracy. The permissions that constitute the subject of our study can be divided into 3 categories according to the abilities that they entail: (1) accessing user privacy, (2) costing money, and (3) other sensitive functionalities. Applications request the permissions to access privacy may leak users' personal information such as location to third parties without being awared. Permissions costing money, such as CALL_PHONE, may be exploited resulting in financial loss to the users. Other sensitive permissions may change settings, start applications on boot, thus possibly wasting phone's battery, and so on. In Table 2, we list the number and percentage of applications declaring each permission in our dataset.

We also parsed the metadata of another 45,811 Android applications from Google Play in May 2014 for our measurements, which assess the description-to-permission fidelity of large-scale applications in Google Play and investigate the correlation between description-to-permission fidelity with application popularity. The metadata include the following features: *category of application, developer of application, number of installations, average rating, number of ratings, descriptions and declared permissions of application.*

5.2 Accuracy Evaluation

5.2.1 Methodology

Whyper studied three permissions: READ_CALENDAR, READ_CONTACTS, and RECORD_AUDIO; Their public results are directly utilized[7] as the ground-truth. The validation set contains around 200 applications for each of the three permissions, where each sentence in the descriptions is identified if revealing the permission by human readers.

[6] `https://github.com/ticcky/esalib`

[7] `https://sites.google.com/site/whypermission/`

Moreover, to assess AutoCog's ability of generalization over other permissions in Table 2, we further randomly select 150 applications requiring each one (except the three permissions previously evaluated in public results of Whyper) as the validation set. For each permission, the complementary set of the validation set is used as the training set to construct the DPR model, which ensures that the validation set is independent of the training set. To get the results of Whyper on other permissions, we leverage the output of PScout [2] and manually extract the semantic pattern set from Android API document[8] following the method presented by Pandita et al. [26]. Whyper's methodology does not work for some permissions such as RECEIVE_BOOT_COMPLETED as they do not have any associated API. To ensure the correctness of our understanding of Whyper's methodology, we contacted Whyper's authors and confirmed our understanding and conclusions. We also tested the system over the applications in their public results and get exactly the same output as those published, further validating the system deployment (source code is released publicly).

Regarding the ground-truth of other permissions that we extend to, we invite 3 participants who are not authors of this paper to read the description and label each sentence as whether or not it suggests the target permission. The description will be classified as "good" when at least two human readers could infer the permission by one sentence in that, or it will be labeled as "bad". Column G_d" in Table 3 is the percentage of "good" descriptions for applications requesting each sensitive permission. The percentage values of "good" descriptions for the 3 permissions GET_TASKS, CALL_PHONE, and READ_CALL_LOG are lower than 10%. We call these permissions rarely described well in descriptions, *hidden permissions*. The scarcity of qualified descriptions leads to the lack of correlated semantic patterns. It would hinder the measurement of description-to-permission fidelity. After removing the 3 hidden permissions, our evaluation focuses on the other 11 permissions.

In training the DPR model, the two thresholds Pre_T and Fre_T balance the performance on precision and coverage of AutoCog. The settings in Table 3 depend on the percentage of applications requesting the permission in the training set. For a permission with fewer positive samples (application requires that permission), each pair of np-counterpart and noun phrase related to it tends to be less dominant in amount, we adjust Fre_T accordingly to maintain the performance on recall. We keep Pre_T high across permissions, which aims at enhancing the precision of detection.

Within the process of deciding if each application description in valuation set warrants permissions, we set the two thresholds Υ=0.8 and Θ=0.67 by empirically finding the best values for them. Low threshold reduces the performance on precision and increasing the threshold excessively causes the increment on false negatives. We set up the threshold Θ lower than Υ, because noun phrases has more diversity in patterns than np-counterparts; phrases containing various numbers of words organized in different orders may express the similar meaning.

Our objective is to assess how closely the decision made by AutoCog on the declaration of permission approaches human recognition given a description. The number of true positives, false positives, false negatives, and true negatives

[8] `http://pscout.csl.toronto.edu/download.php?file=results/jellybean\_publishedapimapping`

Table 3: Statistics and settings for evaluation

Permission	Fre_T	Pre_T	G_d(%)
WRITE_EXTERNAL_STORAGE	9	0.87	38.7
ACCESS_FINE_LOCATION	6	0.85	40.7
ACCESS_COARSE_LOCATION	5	0.8	35.3
GET_ACCOUNTS	4	0.8	26.0
RECEIVE_BOOT_COMPLETED	5	0.85	37.3
CAMERA	3	0.8	48.7
GET_TASKS	3	0.9	2.0
READ_CONTACTS*	3	0.8	56.8
RECORD_AUDIO*	3	0.8	64.0
CALL_PHONE	2	0.8	10.0
WRITE_SETTINGS	2	0.85	44.7
READ_CALL_LOG	3	0.95	6.0
WRITE_CONTACTS	2	0.9	42.0
READ_CALENDAR*	1	0.85	43.6

Hidden permissions are shadowed;
* sampled by around 200 applications, others by 150 applications

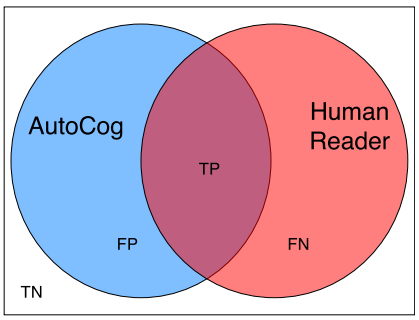

Figure 4: Interpretation of metrics in evaluation

are denoted as TP: the system correctly identifies a description as revealing the permission, FP: the system incorrectly identifies a description as revealing the permission, FN: the system incorrectly identifies a description as not revealing the permission, and TN: the system correctly identifies a description as not revealing the permission. Interpretation of the metrics is shown in Figure 4. Intersection of decisions made by AutoCog and human is true positive. Difference sets between decisions made by AutoCog and human are false positive and false negative, respectively. Complementary set of the union of decisions made by AutoCog and human is true negative. Values of *precision*, *recall*, *F-score*, and *accuracy* represent the degree to which AutoCog matches human reader's recognition in inferring permission by description.

$$Precision = \frac{TP}{TP + FP},$$
$$Recall = \frac{TP}{TP + FN},$$
$$F\text{-}score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall},$$
$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}.$$

5.2.2 Results

Results of our evaluation are given in Table 4. AutoCog matches human in inferring 11 permissions with the average precision, recall, F-score, and accuracy as 92.6%, 92.0%, 92.3%, and 93.2%. As discussed before, Whyper fails to get results for permission RECEIVE_BOOT_COMPLETED. For

Table 4: Results of evaluation

System	Permission	*TP*	*FP*	*FN*	*TN*	*Prec* (%)	*Rec* (%)	*F* (%)	*Accu* (%)
AutoCog	WRITE_EXTERNAL_STORAGE	53	6	5	86	89.8	91.4	90.6	92.7
	ACCESS_FINE_LOCATION	57	3	4	86	95.0	93.4	94.2	95.3
	ACCESS_COARSE_LOCATION	49	1	4	96	98.0	92.5	95.1	96.7
	GET_ACCOUNTS	34	4	5	107	89.5	87.2	88.3	94.0
	RECEIVE_BOOT_COMPLETED	51	6	5	88	89.5	91.1	90.3	92.7
	CAMERA	67	7	6	70	90.5	91.8	91.2	91.3
	READ_CONTACTS	99	5	9	77	95.2	91.7	93.4	92.6
	RECORD_AUDIO	117	10	11	62	92.1	91.4	91.8	89.5
	WRITE_SETTINGS	65	7	2	76	90.3	97.0	93.5	94.0
	WRITE_CONTACTS	57	4	6	83	93.4	90.5	91.9	93.3
	READ_CALENDAR	79	5	6	105	94.0	92.9	93.5	94.4
	Total	728	58	63	936	92.6	92.0	92.3	93.2
Whyper	WRITE_EXTERNAL_STORAGE	11	8	47	84	57.9	19.0	28.6	63.3
	ACCESS_FINE_LOCATION	31	1	30	88	96.9	50.8	66.7	79.3
	ACCESS_COARSE_LOCATION	28	1	25	96	96.6	52.8	68.3	82.7
	GET_ACCOUNTS	9	2	30	109	81.8	23.1	36.0	78.7
	RECEIVE_BOOT_COMPLETED	*Fail to get results*							
	CAMERA	26	4	47	73	86.7	35.6	50.5	66.0
	READ_CONTACTS	89	9	19	73	90.8	82.4	86.4	85.3
	RECORD_AUDIO	105	10	23	62	91.3	82.0	86.4	83.5
	WRITE_SETTINGS	59	24	8	59	71.1	88.1	78.7	78.7
	WRITE_CONTACTS	53	9	10	78	85.5	84.1	84.8	87.3
	READ_CALENDAR	78	15	7	95	83.9	91.8	87.6	88.7
	Total	489	83	246	817	85.5	66.5	74.8	79.9

the remaining 10 permissions, Whyper achieves the average precision, recall, F-score, and accuracy as 85.5%, 66.5%, 74.8%, and 79.9%.

Across the permissions evaluated, the least precision and recall of AutoCog are 89.5% and 87.2%. Even for the cases with low percentage of "good" descriptions and low number of positive samples (permissions GET_ACCOUNT and READ_CALENDAR), our learning-based algorithm and employment of ESA could still get the DPR model aligning with user's recognition well. Whyper could only infer 5 permissions from description (last 5 in Table 4) with both the values of precision and recall higher than 70%. For these permissions, the API documents provide a relatively complete and accurate semantic pattern set. The example patterns such as <"*scan*","*wifi*">, <"*enable*","*bluetooth*">, and <"*set*","*sound*"> could be extracted from the API document of the permission WRITE_SETTINGS. However, Whyper does not perform well on the other 5 permissions. Our understanding is that the patterns extracted from API documents in these cases are very limited to cover the natural-language descriptions with great diversity. For example, the APIs mapped with the permission to write to external storage are related only to download management. Many intuitive patterns, such as <"*save*", "*sd card*">, <"*transfer*", "*file*">, <"*store*", "*photo*"> cannot be found in its API document. It is the same with <"*scan*", "*barcode*">, <"*record*", "*video*"> for camera permission, <"*integrate*", "*facebook*"> (in-app login) for permission to get user's accounts, and <"*find*", "*branch*">, <"*locate*", "*gas station*"> for location permissions. Given Whyper's big variance of performance and our investigation on its source of textual pattern set, we find that suitability of API document to generate a complete and accurate set of patterns varies with permissions due to the limited semantic information in APIs. AutoCog relies on large number of descriptions in training, which would not be restricted by the limited semantic information issue and has stronger ability of generalization over permissions.

Whether or not the API documents are suitable for the evaluated permissions, we note that AutoCog outperforms Whyper on both *precision* and *recall*. Next we discuss several case studies to thoroughly analyze the benefits and limitations of our design.

AutoCog *TP*/Whyper *FN*: The advantage of AutoCog over Whyper on false negative rate (or recall) is caused by: (1) the difference in the fundamental method to find semantic patterns related to permissions, (2) we include the logical dependency between noun phrases as extra ontology. Whyper is limited by the use of a fixed and limited set of vocabularies derived from the Android API documents and their synonyms. Our correlation of permission with noun-phrase based governor-dependent pair is based on clustering results from a large application dataset, which is much richer than that extracted from API documents. Below are 3 examples:
"*Filter by contact, in/out SMS*"
"*Blow into the mic to extinguish the flame like a real candle*"
"*5 calendar views (day, week, month, year, list)*"
The first sentence describes the function of backing up SMS by selected contact. The second sentence reveals a semantic action of blowing into the microphone. The last sentence introduces one calendar application, which provides various views. In our DPR model, the noun-phrase based governor-dependent pairs <*filter, contact*>, <*blow, mic*>, and <*view, calendar*> are found to be correlated to the 3 permissions, READ_CONTACTS, RECORD_AUDIO, and READ_CALENDAR. While the semantic information for the first two sentences cannot be found by leveraging the API documents. For the last one, Whyper could only detect it, as "*view*" and "*calendar*" are tagged with *verb* and *noun*, respectively (both of them are tagged as *noun* here).

AutoCog *TN*/Whyper *FP*: One major reason for this difference in detection is that Whyper is not able to accurately explore the meaning of noun phrase with multiple words. Below is one example:
"*Saving event attendance status now works on Android 4.0*"
The sentence tells nothing about requiring the permission to access calendar. However, Whyper incorrectly labels it as revealing the permission READ_CALENDAR, because it parses resource name "event" and maps it with action "save". AutoCog differentiates the two phrases " event attendance

status" and "event" by using ESA and effectively filters the interference in DPR model training and decision-making.

AutoCog *FN*/Whyper *TP*: This difference is caused by the fact that some semantic patterns implying permissions are not included in the DPR model. Below is one example: "*Ability to navigate to a Contact if that Contact has address*" Whyper detects the word "contact" as resource name and maps it with the verb "navigate". The sentence is thus identified as revealing the permission to read the address book. However, no noun-phrase based governor-dependent pair in our DPR model could be mapped to the permission sentence above, because the pair <*navigate*, *contact*> is not dominant in the training process. The DPR model might not be knowledgeable enough to completely cover the semantic patterns related to the permission. However, the coverage could be enhanced as the size of training set increases.

AutoCog *FP*/Whyper *TN*: In the training process, some semantic patterns, which do not directly describe the reason for requesting the permission in the perspective of user expectation, are selected in the frequency-based measurement. One example is given as:

"*Set recordings as ringtone*"

From this sentence, user could customize her/his ringtone with recording, but it does not directly imply the functionality of recording sound. Our model assigns a high relatedness score between <*set, recording*> and RECORD_AUDIO due to quite a few training samples with related keywords and this permission together. Such cases are due to the fundamental gap between machine learning and human cognition.

AutoCog and Whyper both leverage Stanford Parser [31] to get the tagged words and hierarchal dependency tree. The major cause of the common erroneous detection of two systems (*FP*, *FN*) is the incorrect parsing of sentence by underlying NLP infrastructure, which has been well stated by Pandita et al. [26]. Thus, we would not discuss it in detail given the page limit. As the research in the field of NLP advances underlying NLP infrastructure, the number of such errors will be reduced.

We further list some representative semantic patterns in Table 5, which are found to be closely correlated by our DPR model to the permissions evaluated.

Apart from the accuracy of detection, the runtime latency is a key metric in the practical deployment of AutoCog. We select 500 applications requiring each permission and assess the runtime latency of our system in measuring the description-to-permission fidelity. AutoCog *achieves the latency less than 4.5s for all the 11 permissions.*

5.3 Measurement Results

Our measurements begin with assessing the overall trustworthiness of application market, which is depicted by the distribution of questionable permissions. We utilize AutoCog with the DPR model trained in the accuracy evaluation to analyze 45,811 applications. The training set and dataset for measurements are thus disjoint. The histogram for distribution of questionable permissions is illustrated in Figure 5. Only 9.1% of applications are clear of questionable permissions. Moreover, we measure and observe the negative spearman correlation [19] between the number of questionable permissions of one application by a specific developer with the total number of applications published by that developer (with $r = -0.405$, $p < 0.001$). A possible explanation is that developer publishing more applications

Table 5: Sample semantic patterns

Permission	Semantic Patterns
WRITE_EXTERNAL_STORAGE	<*delete, audio file*> <*convert, file format*> <*download, ringtone*>
ACCESS_FINE_LOCATION	<*display, map*> <*find, branch atm*> <*your, location*>
ACCESS_COARSE_LOCATION	<*set, gps navigation*> <*remember, location*> <*inform, local traffic*>
GET_ACCOUNTS	<*manage, account*> <*integrate, facebook*> <*support, single sign-on*>
RECEIVE_BOOT_COMPLETED	<*change, hd wallpaper*> <*display, notification*> <*allow, news alert*>
CAMERA	<*deposit, check*> <*scanner, barcode*> <*snap, photo*>
READ_CONTACTS	<*block, text message*> <*beat, facebook friend*> <*backup, contact*>
RECORD_AUDIO	<*send, voice message*> <*note, voice*> <*blow, microphone*>
WRITE_SETTINGS	<*set, ringtone*> <*customize, alarm*> <*enable, flight mode*>
WRITE_CONTACTS	<*wipe, contact list*> <*secure, text message*> <*merge, specific contact*>
READ_CALENDAR	<*optimize, time*> <*synchronize, calendar*> <*schedule, appointment*>

Table 6: Correlation between application popularity and the number of questionable permissions and permissions requested. All values are statistically significant with $p<0.001$

	Correlation with application popularity		
Permission Type	$\#install$	$\#rating$	avg_rating
$\#P_q$	-0.106	-0.105	-0.110
$\#P$	0.044	0.050	0.044

are more experienced and likely to be a development team in a company, who is more standardized and better regulated at developing and deploying its mobile software. The above results reflect the severity of the permission-to-description fidelity issue: application publishers, especially the new or personal developer, generally fail to completely cover all the sensitive permissions. The deployment of AutoCog could thus assist developers produce applications with high description-to-permissions fidelity.

We further investigate the correlation between description-to-permission fidelity and application popularity. Application popularity reveals the developers' benefit and users' attitude towards the application, which thus plays a key role in the interaction between users and developers. In our measurements, application popularity is interpreted by the following features: number of installations ($\#install$), number of ratings ($\#rating$), average ratings (avg_rating). Thus, we measure the (spearman) correlation between these three features with the number of questionable permissions ($\#P_q$) and the number of permissions ($\#P$) requested by application, respectively. Table 6 shows that there is a weak positive correlation between application popularity and the number of permissions requested, which is consistent with the results in [5, 13]. It is because that rich functionality

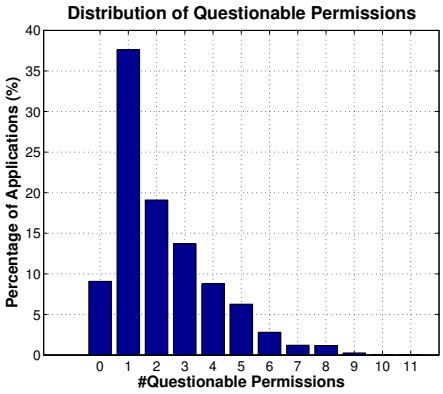

Figure 5: Histogram for distribution of questionable permissions

of application which implies the need of more permissions is the main feature to drive application popularity.

However, we also find the *weak negative correlation* between the number of questionable permissions and the popularity of application. We should note that all the measured results achieve a p-value less than 0.001, which means the statistical significance. We have the following two guesses. First, for the negative correlation, there are a small part of users who are discreet enough or have the professional knowledge to fully understand the security aspects of application metadata [14]. They expect to get permission-related information from the description. Thus the low description-to-permission fidelity negatively affects their decisions of application installation, application assessment, and interest in applications. Secondly, such correlation is weak because most average users cannot tell the questionable permissions based on the description without a tool like AutoCog. Although we could only confirm correlation but not causation here, we expect that wide adoption of AutoCog will help average users to be more security conscious.

6. DISCUSSION

AutoCog measures the description-to-permission fidelity by finding relationships between textual patterns in the descriptions and the permissions. Because of the state-of-the-art techniques used and the new modeling techniques developed, AutoCog achieves good accuracy. Still, AutoCog does have limitations because of the approach it uses and the current implementation.

The models learnt in AutoCog are examples of unsupervised learning, which has the drawback of picking relationships that may not actually exist directly. If a noun phrase appears frequently with a permission, the DPR model will learn that they are actually related. For example, if many antivirus applications use the permission GET_TASKS, the "antivirus" noun may become associated with this permission even if there is no direct relationship between the two. From another perspective though, one could argue that this is even better because AutoCog may be able to extract implicit relationships that human readers may easily miss. Anecdotally, for applications with permission GET_TASKS in our experiments, even if human readers could find only 2% of applications whose descriptions reveal that permission, AutoCog finds 18% of such applications.

For the implementation of AutoCog, we could possibly improve the accuracy by including longer noun phrases and np-counterparts. It is an efficiency-accuracy tradeoff. The evaluation of AutoCog also had some limitations. Manual reading is subjective and the results may be biased. However, given that our readers have a technical background, they may be able to discover many implicit relationships that average users ignore, thus putting up greater challenges for AutoCog. Given that whether a description implies a permission itself is subjective and is consequently lack of ground-truth, manual labeling is the best we can do here.

Malicious developers may provide wrong descriptions to evade this approach. But it will be much easier for even average users to find such mismatch between the app's description and its functionality. And given that most apps are not malicious, such attacks will not affect the training of AutoCog.

7. RELATED WORK

NLP has been widely used in the security area. Potharaju et al. [28] propose an approach to analyzing natural language text in network tickets to infer the problem symptoms and resolution actions. Some efforts have focused on automating mining of network failures from syslogs [29] and network logs [22]. Compared with the network tickets and logs, descriptions of applications have much more complex structures and diverse contents, which largely increases the difficulties of ontology modeling. For example, the developer could choose to use either complete sentences or enumeration lists in description; introduction and contact of company may be included for commercial purpose. There are also approaches using a mix of NLP and learning algorithm to infer specifications from API descriptions, code comments, and formal requirement documents [27]. The methods proposed in these papers require meta-information from source code. Our design only needs the natural language text of descriptions, which is not constrained by the availability of source code and meta-information.

The permission system in Android security framework manages the access of third-party applications to privacy- and security-relevant parts of API. Many previous studies analyze the permission system and resolve the overprivilege issue [2, 11], confused deputy [6, 15, 7] and collusion attack [4]. Moreover, some studies also investigate the effectiveness of permission model [13, 20]. Some researchers have alluded to lack of correlation between permissions and descriptions [3]; however, even if permissions and descriptions do not correlate, our solution can bring an improvement to the current situation. Lin et al. [23] utilize crowdsourcing collect users expectations of the permissions required by application and Han et al. [17] propose a text mining-based similarity measure method to obtain similar security polices among Android applications, which are both complimentary to our work. While the static/run-time analysis of binaries and programming language analysis enable these approaches to detect overprivilege and confused deputy attack, the end user does not have knowledge about why the permission is requested or tools to assess whether applications overstep user expectation. Our system analyzes the descriptions of applications that the end user has direct and easy access to and labels the sentences revealing sensitive permissions, which enables users to know the reason for declaring the permission in the semantic level.

The most relevant work is WHYPER [26], which is the only previous work to our knowledge on bridging the gap between what user expects an application to do and what it really does. Our automatic learning-based approach works directly on large-scale descriptions to select noun-phrase based governor-dependent pairs related to each permission. Thus we would not come across the limitations of WHYPER discussed in Section 2.2.

8. CONCLUSION

In this paper, we propose the system AUTOCOG that measures the description-to-permissions fidelity in Android, i.e., whether the permissions requested by Android applications match or can be inferred from the applications' descriptions. The use of a novel learning-based algorithm and advanced NLP techniques allows us to mine relationships between textual patterns and permissions. AUTOCOG outperforms previous work on both performance of detection and ability of generalization over permissions by a large extent. In inferring eleven permissions by description, our system achieves the average precision of 92.6% and the average recall of 92.0% as compared to previous state-of-the-art 85.5% and 66.5%. Our measurements show a generally weak description-to-permissions fidelity on the Google Play store.

9. ACKNOWLEDGMENTS

This paper was made possible by NPRP grant 6-1014-2-414 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

10. REFERENCES

[1] Android Captures Record 81 Percent Share of Global Smartphone Shipments in Q3 2013. `http://blogs.strategyanalytics.com/WSS/post/2013/10/31/Android-Captures-Record-81-Percent-Share-of-Global-Smartphone-Shipments-in-Q3-2013.aspx`.

[2] K. W. Y. Au, Y. F. Zhou, Z. Huang, and D. Lie. Pscout: analyzing the android permission specification. In *ACM CCS*, 2012.

[3] K. Benton, L. J. Camp, and V. Garg. Studying the effectiveness of android application permissions requests. In *IEEE PERCOM Workshops*, 2013.

[4] S. Bugiel, L. Davi, A. Dmitrienko, T. Fischer, A.-R. Sadeghi, and B. Shastry. Towards taming privilege-escalation attacks on android. In *NDSS Symposium*, 2012.

[5] P. H. Chia, Y. Yamamoto, and N. Asokan. Is this app safe?: a large scale study on application permissions and risk signals. In *ACM WWW*, 2012.

[6] E. Chin, A. P. Felt, K. Greenwood, and D. Wagner. Analyzing inter-application communication in android. In *ACM MobiSys*, 2011.

[7] M. Dietz, S. Shekhar, Y. Pisetsky, A. Shu, and D. S. Wallach. Quire: Lightweight provenance for smart phone operating systems. In *USENIX Security Symposium*, 2011.

[8] Q. Do, D. Roth, M. Sammons, Y. Tu, and V. Vydiswaran. Robust, light-weight approaches to compute lexical similarity. *Computer Science Research and Technical Reports, University of Illinois*, 2009.

[9] W. Enck, P. Gilbert, B. Chun, L. Cox, J. Jung, P. McDaniel, and A. Sheth. Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In *USENIX OSDI*, 2010.

[10] A. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner. A survey of mobile malware in the wild. In *ACM SPSM*, 2011.

[11] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner. Android permissions demystified. In *ACM CCS*, 2011.

[12] A. P. Felt, S. Egelman, and D. Wagner. I've got 99 problems, but vibration ain't one: A survey of smartphone users' concerns. In *ACM SPSM*, 2012.

[13] A. P. Felt, K. Greenwood, and D. Wagner. The effectiveness of application permissions. In *USENIX WebApps*, 2011.

[14] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner. Android permissions: User attention, comprehension, and behavior. In *ACM SOUPS*, 2012.

[15] A. P. Felt, H. J. Wang, A. Moshchuk, S. Hanna, and E. Chin. Permission re-delegation: Attacks and defenses. In *USENIX Security Symposium*, 2011.

[16] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*, 2007.

[17] W. Han, Z. Fang, L. T. Yang, G. Pan, and Z. Wu. Collaborative policy administration. *IEEE TPDS*, 25(2):498–507, 2014.

[18] P. Hornyack, S. Han, J. Jung, S. Schechter, and D. Wetherall. These aren't the droids you're looking for: retrofitting android to protect data from imperious applications. In *ACM CCS*, 2011.

[19] M. G. Kendall. Rank correlation methods. 1948.

[20] K. Kennedy, E. Gustafson, and H. Chen. Quantifying the effects of removing permissions from android applications. In *IEEE MoST*, 2013.

[21] T. Kiss and J. Strunk. Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32(4):485–525, 2006.

[22] C. Lim, N. Singh, and S. Yajnik. A log mining approach to failure analysis of enterprise telephony systems. In *IEEE DSN*, 2008.

[23] J. Lin, S. Amini, J. I. Hong, N. Sadeh, J. Lindqvist, and J. Zhang. Expectation and purpose: understanding users' mental models of mobile app privacy through crowdsourcing. In *ACM Ubicomp*, 2012.

[24] R. Mihalcea, C. Corley, and C. Strapparava. Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI*, 2006.

[25] D. L. Olson and D. Delen. *Advanced data mining techniques*. Springer, 2008.

[26] R. Pandita, X. Xiao, W. Yang, W. Enck, and T. Xie. Whyper: Towards automating risk assessment of mobile applications. In *USENIX Security*, 2013.

[27] R. Pandita, X. Xiao, H. Zhong, T. Xie, S. Oney, and A. Paradkar. Inferring method specifications from natural language api descriptions. In *IEEE ICSE*, 2012.

[28] R. Potharaju, N. Jain, and C. Nita-Rotaru. Juggling the jigsaw: Towards automated problem inference from network trouble tickets. In *USENIX NSDI*, 2013.

[29] T. Qiu, Z. Ge, D. Pei, J. Wang, and J. Xu. What happened in my network: mining network events from router syslogs. In *ACM SIGCOMM*, 2010.

[30] J. C. Reynar and A. Ratnaparkhi. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the fifth conference on Applied natural language processing*, 1997.

[31] R. Socher, J. Bauer, C. D. Manning, and A. Y. Ng. Parsing with compositional vector grammars. In *Proceedings of the ACL*, 2013.

[32] L. K. Yan and H. Yin. Droidscope: seamlessly reconstructing the os and dalvik semantic views for dynamic android malware analysis. In *USENIX Security Symposium*, 2012.

[33] Y. Zhou, Z. Wang, W. Zhou, and X. Jiang. Hey, you, get off of my market: Detecting malicious apps in official and alternative android markets. In *NDSS*, 2012.