Yang and Liang *Cybersecurity* (2018) 1:13
https://doi.org/s42400-018-0011-x

Cybersecurity

RESEARCH Open Access

Automated identification of sensitive data from implicit user specification

Ziqi Yang and Zhenkai Liang*

Abstract

The sensitivity of information is dependent on the context of application and user preference. Protecting sensitive data in the cloud era requires identifying them in the first place. It typically needs intensive manual efforts. More importantly, users may specify sensitive information only through an implicit manner. Existing research efforts on identifying sensitive data from its descriptive texts focus on keyword/phrase searching. These approaches can have high false positives/negatives as they do not consider the semantics of the descriptions. In this paper, we propose S3, an automated approach to identify sensitive data based on users' implicit specifications. Our approach considers semantic, syntactic and lexical information comprehensively, aiming to identify sensitive data by the semantics of its descriptive texts. We introduce the notion *concept space* to represent the user's notion of privacy, by which our approach can support flexible user requirements in defining sensitive data. Our approach is able to learn users' preferences from readable concepts initially provided by users, and automatically identify related sensitive data. We evaluate our approach on over 18,000 top popular applications from Google Play Store. S3 achieves an average precision of 89.2%, and average recall 95.8% in identifying sensitive data.

Introduction

In the cloud era, cloud-based application vendors often collect a large variety of information about users to provide customized service and enhanced experience. Protecting users' sensitive data from being leaked is one of the major challenges in such applications. Recent security incidents of cloud-based services have shown that the privacy of hundreds of millions of users can be easily affected (Yahoo! data breaches; USA Today: Driver's license) in compromised cloud services .

Protecting sensitive user data requires identifying them in the first place, which is the basis of providing stronger but more expensive mechanisms to ensure sensitive data security against a powerful adversary. There are many existing research efforts (Enck et al. 2010; Lu et al. 2015; Rastogi et al. 2013; Zhou and Jiang 2013; Mannan and van Oorschot 2007; Zhou and Evans 2011; Budianto et al. 2014) on protecting sensitive data both on client applications and the cloud platforms. Researchers also develop solutions that utilize trusted environments (Bursztein et al. 2012; Roalter et al. 2013; Oprea et al. 2004; Sharp et al. 2008), such as secure mobile devices, to protect the security of sensitive data from being compromised. Other solutions (Li et al. 2014; Enck et al. 2010; Lu et al. 2015; Rastogi et al. 2013) perform taint analysis based on user-specified sensitive data sources in the mobile platforms to study the data flow of these sources.

However, the sensitivity of data is often *subjective*. Deciding whether a piece of data is sensitive subjects to users' preference and the application context. User's privacy preference may also change overtime. Identifying sensitive data typically needs intensive manual efforts (Nan et al. 2015), which can hardly guarantee accuracy and cannot support large-scale analysis of sensitive data. Therefore, we need a way that can automatically identify sensitive data based on users' subjective requirements.

In this work, we propose a technique to identify sensitive data from implicit user specifications. We focus on mobile applications, but the solutions we develop can be more generally applicable in other scenarios of protecting user privacy in the cloud environment, which we will discuss at the end of the paper. Sensitive data is hardly distinguishable from insensitive data in their technical representations. For example, in an mobile application, the element displaying a user's bank balance is technically represented as the same type of text element displaying

*Correspondence: liangzk@comp.nus.edu.sg
School of Computing, National University of Singapore, Singapore, Singapore

Springer Open

© The Author(s). 2018 **Open Access** This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

the bank name. However, it's easy for the user to know that her bank balance is more sensitive than the bank name, because she understands the meaning of the data or its surrounding descriptive text. Therefore, instead of analyzing source code of applications, it is more accurate and efficient to identify sensitive data from the user interface (UI), and understand the semantic meaning of data or its surrounding descriptive text.

Several solutions have been proposed to identify sensitive data using the descriptive text in mobile applications. Supor (Huang et al. 2015) identifies sensitive input data of Android applications by keyword based searching on descriptive texts. UIPicker (Nan et al. 2015) utilizes SVM (Support Vector Machine) to learn sensitive descriptive texts with sensitive keywords as features. AutoCog (Qu et al. 2014) identifies the real permissions an Android application requires from its descriptions on Google Play, by analyzing the semantic meaning of noun phrases, verb phrases and possessives. Whyper (Pandita et al. 2013) considers both actions and noun phrases to further increase the accuracy. However, all the existing approaches are based on only key word/phrase/counterpart searching, with no complete semantic information considered. For instance, all of them incorrectly classify the sentence "*Facebook will not save your password*" as sensitive because of the detection of a sensitive phrase "save your password", though it is only a normal claim message. Moreover, none of the prior work takes into account the flexible user requirements.

In this paper, we propose a novel technique, S3[1], to identify sensitive data. S3 aims to understand users' preferences by extracting the semantic concepts from a set of user-provided texts, and identifies unseen sensitive data with a learning-based approach. Instead of outputting a Boolean result in prior work, S3 produces a reference probability of a text being sensitive to make the measurement controllable by setting a threshold in different strictness levels. Besides, S3 classifies sensitive data as multiple categories such as credential data, profile data and financial data, in a more fine-grained way, so that users and developers are able to choose different categories of sensitive data on demand for further protection.

Contributions.

- To the best of our knowledge, S3 is the first automated approach to precisely identify sensitive data by analyzing its semantic meaning on a large scale. S3 reduces much manual effort of identifying sensitive data for further protection and research on it.
- S3 supports flexible user requirements in defining sensitive data. It enables users to define sensitive data on demand by providing initial concepts. Then S3 is able to automatically identify unseen sensitive data by learning from the concepts.
- We conduct a series of evaluation, and compare S3 with existing approaches. Evaluation results show that S3 is able to identify sensitive data with high precision and recall, and can correctly identify instances which are not handled in existing approaches.

Overview

In this section, we introduce our motivation, and analyze the challenges faced by sensitive data identification. We then introduce techniques used in natural language processing (NLP) as a background.

Motivation

Many existing research efforts (Enck et al. 2010; Lu et al. 2015; Rastogi et al. 2013; Zhou and Jiang 2013; Mannan and van Oorschot 2007; Zhou and Evans 2011; Bursztein et al. 2012; Roalter et al. 2013; Oprea et al. 2004; Sharp et al. 2008) have been proposed on sensitive data protection and taint analysis in mobile and web applications. In particular, many existing work (Li et al. 2014; Enck et al. 2010; Lu et al. 2015; Rastogi et al. 2013) studies the security topics on predefined sensitive data sources. For example, one can track the privacy disclosures of sensitive user inputs to different sinks with proper static or dynamic taint analysis. With static program analysis, one can also find the vulnerabilities in mobile applications that may disclosure sensitive user inputs to public or attacker controlled output. Secure display and input (Li et al. 2014) provides an end-to-end channel that ensures sensitive display and input can never be accessed by untrusted Android drivers.

However, identifying these sensitive data in existing work requires much manual effort. Furthermore, manual identification of sensitive data cannot guarantee the accuracy and large-scale deployment. Therefore, a systematic automated approach to identify sensitive data is important and required to deploy existing research work on sensitive data protection efficiently on a large scale. It is necessary to automatically identify sensitive data for further protection or analysis of existing work. Hence, it helps reduce the burden of manual identification, and increases the accuracy as well.

Another important fact is that sensitive data are quite subjective to users' preferences and application scenarios. For instance, a banking application is most likely to handle users' sensitive financial data, while a social network application contains mostly sensitive profile data. Users might have different preferences in terms of sensitive data as well. Existing work treats all sensitive data as one category which is not practical to users. It might include data which are not important to some users, and meanwhile miss data which are sensitive for other users. Such inaccurate identification of sensitive data without notions of

users' preferences could either bring about users' additional efforts to handle unimportant data, or expose user's real sensitive information unprotected. Therefore, it is important to identify sensitive data in a more flexible and on-demand way.

Sensitive data (e.g., login input box, shopping history, and profile data) in a UI widget is usually surrounded or embedded with a descriptive text (Huang et al. 2015; Nan et al. 2015), indicating its functionality. Figure 1 shows two mobile screens that contain some critical sensitive data in the Amazon Shopping application (Amazon shopping application). Figure 1a requires the user to input the credit card information to accomplish the payment process. Both input boxes are associated with descriptive text *"Name on card"* and *"Card number"*. In Fig. 1b, user's login profile data is listed on the screen with corresponding descriptive text as well. Such descriptive text in mobile applications is usually short, and well-spelled. By understanding such descriptive text, one is able to identify the sensitive data for further protection or analysis. Supor (Huang et al. 2015) proposes an effective solution to locate the sensitive data based on the location of corresponding descriptive text. However, it only performs simple keyword based searching on identifying the descriptive text which limits its capability of handling unseen text. Therefore, the core part in identifying sensitive data is to understand the semantic meaning of surrounding descriptive text, which motivates our work.

Challenges in Sensitive Data Identification.

Semantic Understanding. The data sensitiveness is highly dependent on its semantic meaning. For instance, the sentence "*Facebook will not save your password.*" does not indicate sensitive data, because it is just a declaration text showing Facebook will not violate users' privacy. Prior work (Huang et al. 2015; Nan et al. 2015; Qu et al. 2014; Pandita et al. 2013) incorrectly identifies this text as sensitive because of the detection of a key phrase "save your password". Consider two sentences, "*Register account*" and "*Account registered*". The former one describes a sensitive operation *requesting* information, and the latter one is a hint text *confirming* that the account has already been registered. Existing work cannot distinguish the sensitiveness of the two sentences, because they are composed of same sensitive keywords "register" and "account". Another representative example is "*Log in*" and "*Logged in*". The first one describes a sensitive operation but the second one is a normal message indicating the user has already logged

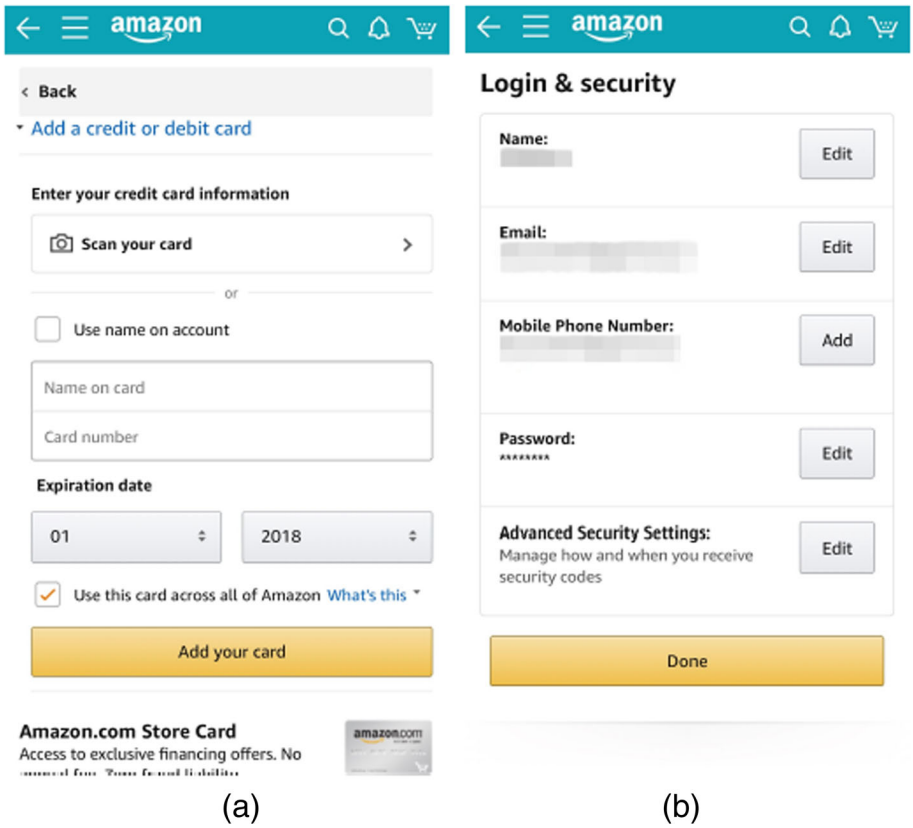

Fig. 1 Examples of users' sensitive data in mobile applications. **a** User's credit card. **b** User's login profile

in. Both of them have the same words, regardless of their forms, but their sensitiveness are significantly different, because of the different part of speech of "log". Therefore, to give a more accurate measurement of the sensitiveness, we have to learn its semantic meaning.

Implicit Specification. Sensitive data are subjective to users' preferences and application contexts. It is preferable to identify sensitive data in a more flexible way, by allowing users to customize the classification of sensitive data they are really cared about. However, it is sometimes intractable for users to explicitly describe/define what data are sensitive based on their implicit preferences. For example, one may want to protect his private profile information (e.g., Fig. 1b), and he may have some notions of it such as first/last name, home address, and phone number. Nonetheless, these notions are far from sufficient to define profile data. It is challenging to understand the implicit meaning of users' preferences from these notions, and further infer more related sensitive data belonging to the same "category". For example, by understanding the semantic meaning of the user's preference on "profile data", it is possible to further automatically classify his age and gender as sensitive.

NLP Background

The semantic meaning of texts is critical for correctly identifying sensitive data. To understand the semantics of a text, we need NLP techniques to process it. With advance of existing NLP techniques, the grammatical structure of a natural language sentence can be parsed accurately. We next briefly introduce the key NLP techniques used in our work.

Parts Of Speech (POS) Tagging (Toutanova et al. 2003; Klein and Manning 2002). It is also called "*word tagging*", "*grammatical tagging*" and "*word-category disambiguation*". POS tagging is able to identify the part of speech (such as nouns and verbs) a particular word in a sentence belongs to. Current state-of-the-art approaches have been shown to achieve 97% (Manning et al. 2014) accuracy in classifying POS tags for well-written news articles.

Named Entity Recognition (Finkel et al. 2005). It is also known as "*entity identification*" and "*entity extraction*", and works as a subtask of information extraction. These techniques are able to classify words in a sentence into predefined categories such as names, quantities, and expressions of time.

Phrase and Clause Parsing. It is also known as "*chunking*". This technique divides a sentence into a constituent set of words (or phrases) that logically belong together (such as a Noun Phrase and Verb Phrase) to analyze the syntax of the sentence. Current state-of-the-art approaches can achieve around 90% (Manning et al. 2014) accuracy in classifying phrases and clauses over well-written news articles.

Syntactic Parsing (Jurafsky and Martin 2000). It generates a parse tree of a sentence showing the hierarchical view of the syntax structure for the sentence. By traversing the parse tree, we are able to identify target phrases (such as noun phrases and verb phrases) and POS tags.

S3 Design

In this section, we present the design of S3. We first give an overview. We then introduce the core components. They analyze the topic of a preprocessed text, and decides its sensitiveness.

Approach Overview

The overall architecture of S3 is illustrated in Fig. 2. S3 takes a raw text as input. *Preprocessor* processes this text to generate an intermediate structure. The intermediate structure is a data structure holding all the required information for further analysis. It contains the syntax information and other information such as noun phrases, verb phrases, POS tags, etc. Then S3 analyzes its topic by

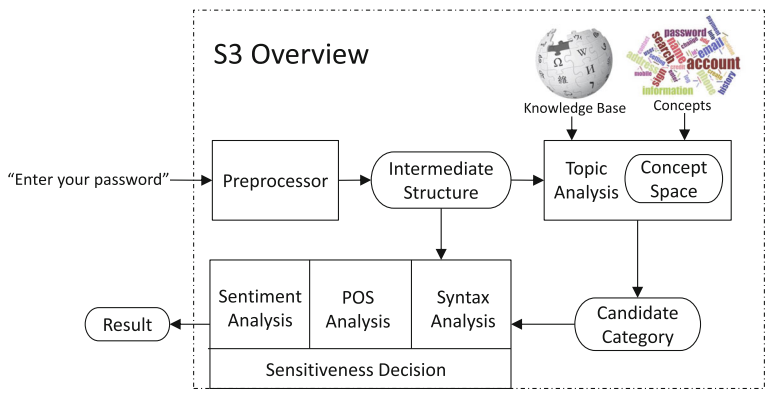

Fig. 2 Overall architecture of S3

extracting its semantic meaning, and produces a candidate sensitive category. Finally, S3 decides its sensitiveness by analyzing its syntax, POS, and sentiment information.

Each sensitive category is represented as a set of vectors, called *concept space* in the following, constructed from concept words/phrases provided by users. The knowledge base is a large corpus of texts, from which S3 is able to identify the semantic relation of unseen texts and the concept space, and thus determines its topic.

Preprocessing

The NLP techniques we have discussed above are used as a preprocessor in S3 to accept the raw natural-language sentences as input and produces an intermediate structure for further analysis. It uses standard NLP techniques to perform text splitting, stopword removal, phrase collection, modifier extraction, lemma recovery, part of speech tagging, and syntactic parsing. Figure 3 gives an illustrating example of partial preprocessing result. The sentence node is labeled as *S*. It is the child of the root node. The interior nodes of the tree are labeled by non-terminal tags (e.g., verb phrases *VP* and noun phrases *NP*). The leaf nodes are labeled by terminal tags (e.g., pronouns *PRP$* and nouns *NN*). In summary, Table 1 lists the members of the intermediate structure of a text after preprocessing.

Text Splition. A piece of text could consist of one or more sentences. In our approach, we use Stanford Parser (Manning et al. 2014) to split a piece of text into mupltiple sentences if any. We empirically observe that the descriptive texts of sensitive data in widgets of mobile UI are short and are usually composed of a single sentence/phrase. For example, A noun phrase "*Your password*" and a sentence "*Enter your password*" describe a password input box. Therefore, a text containing multiple sentences is not likely to be the descriptive text of sensitive data. In our approach, for a coming text, preprocessor will return

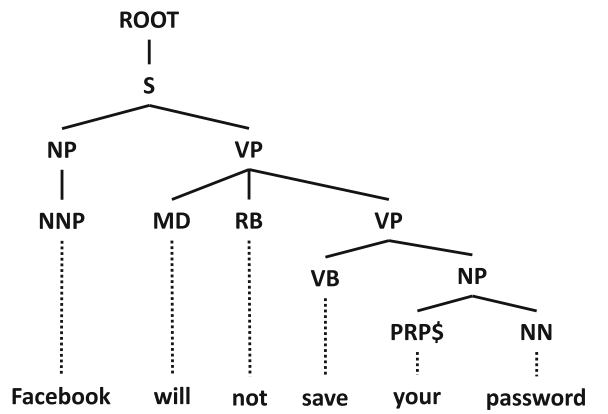

Fig. 3 Preprocessing of a sample sentence

Table 1 Members of the intermediate structure

Item	Description
Noun Phrase List	Reflect the content of the data it describes
Verb Phrase List	Reflect the operation it guides users to perform
Modifiers	Adjectives of nouns in noun phrases
Lemma List	Mapping of original word to its lemma form
POS Tag List	Mapping of a word to its pos tag
Parse Tree	Hierarchical view of the syntax structure for the sentence

identification result of insensitive text if it contains more than one sentence.

Stopword Removal. Stopwords are some of the most common words like "the", "these", "a". Preprocessor removes stopwords because they offer no meaningful help in further analysis.

Named Entity Removal. Sometimes a sequence of words correspond to the name of entities that have a specific meaning collectively (Pandita et al. 2013). For example, the words/phrases "*Google*", "*Facebook*" and "*Google Play*" are the names of companies/services. Further analysis of such phrases would not bring any semantic value. Therefore, we identify such phrases and annotate them as single lexical units. We achieve so using Stanford Parser (Manning et al. 2014).

Phrase Collection. For a single sentence/phrase, preprocessor extracts required information to construct an intermediate structure for further analysis. We have observed that the descriptive texts of input and output data have regular features that help users understand the purpose of the description. They can be summarized as follows.

1. **Descriptive texts reveal the content of surrounding data**. For example, the sentence "*Your password*" describes a password input box. The sentence "*First Name*" describes a first name input box or a field displaying a user's first name.
2. **Descriptive texts guide users to do some operation**. For example, Texts "*Log in*", "*Sign up*" and "*Register an account*" describe operation buttons which conduct users to login, signup and register an account.

We observe that descriptive texts represent the content of surrounding data as noun phrases, and use verb phrases to represent actions/operations. Based on the observation, preprocessor collects all the noun phrases and verb phrases of a sentence into the intermediate structure using phrase and clause parsing as described in "NLP Background" section.

Modifier Extraction. Noun phrases often contain adjectives modifying the nouns. Such modifiers are able

to affect the sensitiveness of a sentence. For example, A noun phrase "*Email address*" describes sensitive information (email address) while another noun phrase "*Invalid email address*" is just a hint sentence showing that the user has typed an invalid email address. In this instance, the modifier "Invalid" reduces the sensitiveness of the sentence. Based on this observation, preprocessor extracts adjectives from the noun phrases and save them in the intermediate structure.

Lemma Recovery. This process is for recovering inflected (or sometimes derived) words to their lemma, base or root form. Lemma recovery makes words such as "accounts" match to the single common lemma "account". Lemma recovery can greatly improve the results of semantic analysis because it transforms the same meaning words to the same word and thus eliminates deviation. Preprocessor gets the lemma form of each word in a sentence using Stanford Parser (Manning et al. 2014) and stores the (word, lemma) pairs in the intermediate structure.

Part of Speech Tagging. The part of speech of a word indicates the role it plays in a sentence. We observe that POS information also effects the sensitiveness of a sentence especially for verb phrases. For example, The verb phrases "*Log in*" describes a sensitive operation but the verb phrase "*Logged in*" is just a hint sentence showing that the user has already logged in. Even though the two verbs have the same lemma form "log", different parts of speech effect the sensitiveness siginificantly. Preprocessor tags each word in a sentence with part of speech and stores the (word, pos) pairs in the intermediate structure.

Syntactic Parsing. Preprocessor generates a parse-tree structure for a sentence based on its grammar structure. The parse tree of a sentence shows the hierarchical view of the syntax structure for the sentence. For example, the parse tree of sentence "*Facebook will not save your password*" is illustrated in Fig. 3. The sentence node is labeled as *S*. It is the child of the root node. The interior nodes of the tree are labeled by non-terminal tags (*e.g.,* verb phrases *VP* and noun phrases *NP*). The leaf nodes are labeled by terminal tags (*e.g.,* pronouns *PRP$* and nouns *NN*). Preprocessor saves the parse tree in the intermediate structure for further analysis.

Topic Analysis

In this part, S3 analyzes the semantic information of a text to produce a candidate category of sensitive data.

Descriptive texts describe the content of surrounding data or some operation/action they guide users to do. Such content and opertions/actions are noted as *topics* in this paper. The topic of a sentence can help S3 classify it to a more specific category of sensitive data. As discussed "Preprocessing" section, noun phrases in a sentence relate to the content of surrounding data and verb phrases relate to actions. Therefore they are used to analyze the topic of a sentence. Both noun phrases and verb phrases in a sentence are extracted by preprocessor in the intermediate structure. The topic of a sentence is reflected by the semantic meaning of noun phrases and verb phrases (*e.g.,* the noun phrase "*Your password*" describes credential related input box and the verb phrase "*Sign up*" describes account related action). In S3, the semantic meaning of both phrases is obtained by measuring the semantic distance between the target phrase and each sensitive category in a vector space, where each category has its own cluster, referred to as *concept space*. The closest sensitive category is chosen to classify its sensitiveness.

Concept Space. We use vector representations of words (Mikolov et al. 2013) to create a domain of sensitive category to represent *concept space* in our approach. The intuition is to cluster closely related sensitive words/phrases in the vector space, so that we are able to classify unseen texts based on the clusters. Such words and phrases in the clusters are called *concepts*, provided by users or developers. For example, one may create a category "Credentials" by feeding concepts of "username", "password" and "pin code". S3 then constructs a concept space based on them for this category. Vector representation of words takes as its input a large corpus of text as knowledge base and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located in close proximity to one another in the space. However, vector representation of words can only encode a single word to a vector, and thus is not applicable to phrases (multiple words). In our approach, we encode a phrase using the mean of each word vector, thus treating each word identically important in a phrase.

An unseen word/phrase is classified to a sensitive category based on K-Nearest Neighbors (KNN) algorithm (Cunningham and Delany 2007). It first transforms the word/phrase to its vector representation, and then calculates K nearest vectors under a predefined threshold of similarity distance in the concept space of each category. A word/phrase is classified to the *i-th* sensitive category if K_i is maximum, and assigned with the maximum similarity score in the concept space as its probability. If no neighbors are found, this word/phrase is classified as insensitive. The similarity distance between two vectors is measured using cosine similarity (Gabrilovich and Markovitch 2007), which is a standard way of quantifying the similarity between two documents in the vector space in document retrieval (Steinbach et al. 2000). In order to improve the result of classification, S3 uses the lemma form of each word in noun phrases and verb phrases. The lemma forms can be obtained in the intermediate structure.

In total, for each category, we define concept space of noun phrases, of actions (verbs), and of single actions. Noun phrases in a descriptive text indicate the content of the surrounding data. They can also be dominated by actions, and such actions could affect the sensitiveness significantly. For example, in the sentence "*Enter your password*", the noun phrase "your password" is dominated by an action "Enter", and it describes sensitive data—a password input box. Nonetheless, in the sentence "*Forgot your password*", the noun phrase "your password" is also dominated by an action "Forgot", but it does not describe any sensitive data. Therefore, it is necessary to classify the sensitiveness by taking actions into consideration. There are also sentences containing only a single action in the descriptive texts, for example, "*Log in*". Some sentences are composed of an action and a named entity, but the entity is removed in the preprocessing phase, and thus only a single action is left. For example, in the sentence "*Pay with Paypal*", the noun phrase "Paypal" is a named entity and thus removed, but its dominant action "Pay" is sensitive. We also define concept of single actions to handle such cases. With the three types of concept space, S3 is able to identify the topic of a sentence.

Modifier Analysis. The semantic meaning of concepts are affected by the words modifying it. Specifically, a noun phrase often contains adjectives that can affect its sensitiveness. For instance, the sentence "*Email address*" describes sensitive profile information but the sentence "*Invalid email address*" is a normal hint message. Therefore it is necessary to analyze the sentiment of the adjectives to improve the accuracy. In our approach, we use SentiWordNet (Baccianella et al. 2010) to give a sentiment score of an adjective. A negative adjective is assigned a negative value. The more negative the word is, the larger the absolute value of its sentiment score is. S3 first collects adjectives in a noun phrase using POS information in the intermediate structure. Then it adds the sentiment score to its probability of being sensitive if the sentiment score is negative and thus reduces its probability.

Sensitiveness Decision

Topic analysis, giving only candidate sensitive category, is not sufficient to determine the sensitiveness. For instance, the sentence "*Register account*" describes a sensitive action but the sentence "*Account registered*" is an insensitive message showing that the account has already been registered. However they are both classified as the same sensitive category because they consist of same words in lemma form. The sentence "*Log in*" describes a sensitive action while the sentence "*Logged in*" is insensitive although it includes sensitive word "log". In the sentence "*Login failed*", the negative sentiment of this sentence makes it insensitive although it has sensitive action "login". Therefore, despite topic analysis, S3 also performs syntax, lexical and sentiment analysis to finally determine the sensitiveness.

Syntax Analysis. Empirically, we observe noun phrases and verb phrases in sensitive descriptive texts usually have fixed syntactic patterns, and we note such nodes a *Candidate Block* (*CB*) in the parsing tree. We summarize three syntactic patterns with *CB* notations of a descriptive text as follows:

- **Noun phrase only** (CB_{NP}). The noun phrase directly indicates the content of the surrounding data. For instance, "*Your password*" indicates a password input box.
- **Verb only** (CB_A). The action (verb) indicates some operation. For instance, "*Log in*" and "*Register*" describe sensitive account operations. "*Pay with Paypal*" is also a CB_A, as shown in Fig. 4c, where "Paypal" is removed as a name of entity.
- **Verb phrase (verb+noun phrase) (CB_{ANP})**. In the parsing tree of a sentence, if a noun phrase node has

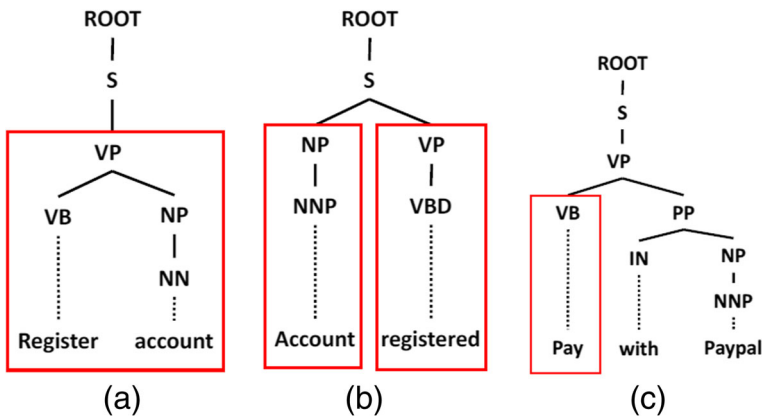

Fig. 4 Syntax analysis illustration. The red rectangles indicate *CBs*. **a** "*Register account*". **b** "*Account registered*". **c** "*Pay with Paypal*"

an ancestor verb phrase node (*VP*), we say the noun phrase is dominated by the action in the *VP*. For instance, in "*Register account*", the noun phrase node "account" has a father *VP* node of action "Register" as shown in Fig. 4a, then the noun phrase is dominated by the action.

In the case of CB_A and CB_{NP} co-existing in a text without forming a CB_{ANP}, we empirically defines the priority of CB_{NP} is higher than that of CB_A. For example, in the case of "*Account registered*" as shown in Fig. 4b, the noun phrase "account" is the emphasis of the sentence, and the action "registered" modifies it.

After S3 collects all the *CB*s, it checks if their ancestor nodes or sibling nodes contain verb phrases. If verb phrases are found, the sentence is identified as insensitive. This is because the surrounding verb phrases can reduce the sensitiveness of a *CB* significantly. In Fig. 4b, the CB_{NP} "Account" has a sibling verb phrase "registered". Therefore the meaning of this sentence is to state the noun phrase is operated by the action and thus the sensitiveness of the noun phrase is reduced to insensitive by the action. In Fig. 4a, the CB_{ANP} has no surrounding nodes containing verb phrases, so the whole sentence is sensitive.

POS Analysis. For sentences containing only a CB_A , e.g., *"Log in"* and "*Logged in*", POS affects the sensitiveness of the CB_A significantly. In our approach, we assume only the base form (noted as *VB* in the parse tree) and non-3rd person singular present (noted as *VBP* in parse tree) of a verb remain the sensitiveness. Other forms of a verb will reduce the sensitiveness of the CB_A to insensitive. For example, the word "Logged" is tagged as *VBN* (past participle) in the sentence "*Logged in*". The sensitiveness of this sentences is reduced to insensitive. S3 checks POS of the action in a CB_A of a sentence to revise its sensitiveness.

Sentiment Analysis. We observe that negative sentiment of descriptive texts makes it insensitive (e.g., *"Login failed"*). We have tried Stanford Sentiment Analysis (Socher et al. 2013) but it does not produce satisfactory results of identifying the sentiment of texts in our problem domain. We have observed that most of the negative descriptive texts include some common keywords like "fail", "no", "error", etc. In our approach, we make a list of such negative words. S3 performs a keyword-based searching in a sentence to analyze its sentiment. A sentence is negative if it contains any of the negative keywords.

Implementation

In this section, we describe the details of our implementation of S3, including the frameworks and tools we built upon.

We implemented the preprocessor based on Stanford CoreNLP (Manning et al. 2014) to generate the intermediate structure, which is a state-of-the-art suit of core NLP tools including Stanford Parser from Stanford. It can give the base forms of words, their parts of speech, whether they are names of entities, mark up the structure of sentences in terms of phrases and word dependencies. It has multiple annotators to indicate different NLP tools. We use annotators in the order of "*tokenize, cleanxml, ssplit, parse, pos, lemma, ner*" to remove XML tokens from a document, split sentences, tokenize a sentence, give the lemma form and part of speech of a token, label named entities, and generate the parse tree. We make a list of stopwords to help preprocessor remove stopwords in a sentence.

To map words to the concept space, we implemented Stanford GloVe (Pennington et al. 2014) for vector representation of words. GloVe is the state-of-art unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. We use English Wikipedia 2014 (Wikipedia) and Gigaword version 5 (LDC: English gigaword fifth edition) as the base corpus. Each word is represented as a vector of 300 dimensions.

To analyze the sentiment of a sentence, we make a list of negative keywords to detect negative sentences. The list we use in this paper is "fail", "error", "wrongly", "wrong", "invalid", "incorrect", "miss", "no", "not".

Excluding the underlying libraries, our prototype of S3 consists of 1156 lines of code in Java.

Evaluation

In this section, we present the evaluation of S3. Given a piece of text, S3 classifies it to some sensitive category with a probability. We first introduce the experiment setup of evaluation. Then we analyze the results of evaluation, and compare S3 with related work. Finally we analyze the causes of producing false positives and false negatives.

Evaluation Setup

We evaluate S3 on Android applications. We get all the text resources of an Android application using *decompiling* technique. In this paper, we use texts from a snapshot of popular Android applications. The app data set was generated from the official Google Play Store in November 2016. It contains the top 500 free applications in each category (34 categories totally). Except some connection errors occurred in the downloading process, we collected 18,681 applications totally. For each application, we extract texts from */res/values/strings.xml* file after decompiling it. We remove non-English texts from all the 18,681 applications and finally get 1,741,143 distinct English texts. We sort them based on the frequency of each text appearing in all the applications. In our evaluation, we manually define 7 sensitive categories. For each

category, we define concepts for noun phrases, actions and single actions respectively. Information of the 7 categories is illustrated in Table 2.

We first manually annotate the top 5152 frequent texts using the listed categories. In our evaluation, we invite five volunteers to annotate these texts independently. A text is annotated as one category only if at least three volunteers label the text as the same category. Then S3 is applied on these texts to output results under different thresholds of similarity distance from 0.5 to 1.0 with 0.05 as interval. For each threshold, we measure the number of *true positives (TP)*, *false positives (FP)*, *true negatives (TN)* and *false negatives (FN)*, which are illustrated as follows:

- **TP**: A text which S3 correctly identifies as sensitive (category).
- **FP**: A text which S3 incorrectly identifies as sensitive (category).
- **TN**: A text which S3 correctly identifies as not sensitive (category).
- **FN**: A text which S3 incorrectly identifies as not sensitive (category).

In statistical classification (Olson and Delen 2008), *Precision* is defined as the ratio of the number of true positives to the total number of items reported to be true, and *Recall* is defined as the ratio of the number of true positives to the total number of items that are true. *F-score* is defined as the weighted harmonic mean of Precision and Recall. *Accuracy* is defined as the ratio of sum of true positives and true negatives to the total number of items. Higher values of precision, recall, F-Score, and accuracy indicate higher quality of S3 to identify sensitive data. Based on the total number of TPs, FPs, TNs, and FNs, we compute the precision, recall, F-score, and accuracy of S3 in identifying sensitive texts as follows:

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

Table 2 Categories of sensitive data

Category	# CoNP	# CoA	# CoSA
Account	5	16	6
Calendar	4	5	0
Credential	16	18	0
Finance	30	10	4
Profile	45	21	0
Search & History	6	6	2
Setting	8	9	1

#CoNP: Number of noun phrases; #CoA: Number of actions; #CoSA: Number of single actions

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$F\text{–}score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{3}$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{4}$$

Results

In this section, we describe the evaluation results and compare S3 with related work. We first measure the effectiveness of S3 under different thresholds in identifying sensitive texts to find the optimal threshold. Then we analyze in detail the effectiveness under the optimal threshold. Finally, we compare S3 with other approaches.

Threshold Setting

Errors of S3 come from false positives and false negatives. S3 seeks to achieve higher performance than prior work by reducing false positives and false negatives. However, to choose the optimal threshold in the trade-off of false positives and false negatives, we seek less false negatives than false positives. This is because false positives identify normal data as sensitive, and thus cause over protection, while false negatives leave sensitive data unprotected, and cause more serious consequences, e.g., data exposed to attackers. In statistics, *Recall* can reflect the measure of false negatives and *Precision* reflects false positives. Therefore we seek higher recall value than precision value in this paper.

The threshold controls the relatedness measure between a target noun phrase and action with concept spaces. A higher threshold indicates that the target is classified into a concept space only with closer relation with the concept space. Evaluation results differ under different thresholds as illustrated in Fig. 5. We compute average precision, recall, F-score and accuracy of the 7 categories for each threshold. The threshold ranges from 0.5 to 1.0 with 0.05 as interval. The results show that as threshold increases, precision first increases sharply before threshold 0.7 and then increases smoothly. Recall first increases smoothly before threshold 0.7 and then decreases sharply. The reason of such trend is that a higher threshold means S3 identifies a text as sensitive more strictly which causes less false positives but more false negatives. The accuracy differs more smoothly than precision, recall and F-score. This is because the number of negative samples (4588 identified by human) is much larger than the number of positive samples (564 identified by human) in 5152 texts. Therefore the fluctuation between true positives and true negatives is small and thus affects accuracy little. We can conclude from the results that under threshold 0.70, recall

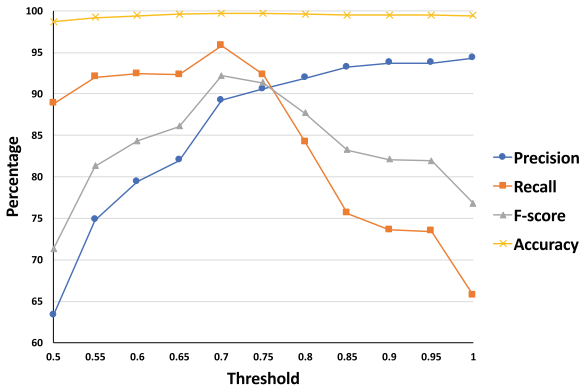

Fig. 5 Evaluation results under different thresholds

(95.8%) gets its maximum value and both F-score (92.2%) and accuracy (99.7%) get the maximum value as well.

Effectiveness Analysis

In this section, we evaluate the effectiveness of S3 in identifying sensitive texts. We take the optimal threshold 0.7 as an example to describe the evaluation results. Table 3 shows the evaluation results under threshold 0.7. Column "Category" lists names of the 7 predefined categories of sensitive data. Column "H_I" lists the number of texts identified as corresponding category of sensitive data by human users. Column "M_I" lists the number of texts identified as corresponding category of sensitive data by S3. Columns "*TP*", "*FP*", "*TN*" and "*FN*" list the number of *true positives*, *false positives*, *true negatives* and *false negatives* respectively. Columns "P(%)", "R(%)", "F_S(%)" and "*Acc*(%)" list the percentage of *Precision*, *Recall*, *F-score* and *Accuracy* respectively. The evaluation results show that S3 effectively identifies and classifies sensitive texts out of top 5152 frequent texts with average precision, recall, F-score, and accuracy of 89.2%, 95.8%, 92.2% and 99.7% respectively. If we treat sensitive data as one category, S3 achieves precision, recall, F-score, and accuracy of 87.1%, 96.6%, 91.6%, and 98.1% respectively. Evaluation results show that S3 produces less false negatives than false positives in most categories which is in line with our expectations. We will discuss the reasons of producing false positives and false negatives in "FP/FN Analysis" section.

Comparison with related work

In this section, we compare S3 with related approaches in identifying sensitive texts. Table 4 shows the comparison results among different approaches in identifying sensitive texts. We select eight typical instances to present the comparison process.

For the sentence "*Enter your password*", it describes an input box receiving user's password. It contains sensitive keywords "Enter" (an action) and "password" (a noun phrase). All the approaches are able to correctly identify this sentence as sensitive. Moreover, S3 can classify it as category "Credential". The second sentence "*Facebook will not save your password*" also contains sensitive keywords "save" (an action) and "password" (a noun phrase). However it is only a normal hint sentence and describes no sensitive data. Both Supor and UIPicker incorrectly identify it as sensitive because it contains sensitive keywords. Whyper also fails because the sentence contains a sensitive noun phrase "your password" with dominant action "save". AutoCog fails as well because the sentence has a sensitive verb phrase "save your password", a sensitive noun phrase "your password". However even though the sentence has a sensitive noun phrase with dominant action (make up a CB_{ANP}), it does not guarantee that it is the point of the sentence. S3 correctly identifies it because it finds that the CB_{ANP} is dominated by "Facebook will" (has an ancestor node of verb phrase) so that the CB_{ANP} is not the point of the sentence and thus the whole sentence is not sensitive.

Table 3 Evaluation results under threshold 0.7

Category	H_I	M_I	*TP*	*FP*	*TN*	*FN*	P(%)	R(%)	F_5(%)	*Acc*(%)
Account	66	66	63	3	5083	3	95.5	95.5	95.5	99.9
Calendar	18	19	17	2	5132	1	89.5	94.4	91.9	99.9
Credential	77	87	75	12	5063	2	86.2	97.4	91.5	99.7
Finance	83	103	81	22	5047	2	78.6	97.6	87.1	99.5
Profile	200	232	193	39	4913	7	83.2	96.5	89.4	99.1
Search & History	42	41	39	2	5108	3	95.1	92.9	94.0	99.9
Setting	78	78	75	3	5071	3	96.2	96.2	96.2	99.9
Average	-	-	-	-	-	-	89.2	95.8	92.2	99.7
One Category*	564	626	545	81	4507	19	87.1	96.6	91.6	98.1

H_I: Number of texts identified by human as sensitive (category); M_I: Number of texts identified by S3 as sensitive (category); *TP*: Number of true positives; *FP*: Number of false positives; *TN*: Number of true negatives; *FN*: Number of false negatives; *P*: Precision; *R*: Recall; F_5: F-score; *Acc*: Accuracy; *: The last row is computed by treating all sensitive texts as one category

Another comparison example is sentences "*Register Account*" and "*Account registered*". Both of the two sentences have sensitive keywords "Register" and "Account", but the meanings of them are significantly different. The former one describes an account registration manner but the latter one is a hint message saying that the account is already registered. All the related approaches can correctly identify the former sentence but fail for the latter sentence. As we are not sure if an action check is compulsory or not in Whyper, we here assume it is not compulsory so that Whyper correctly identifies the sentence. S3 first identifies the sensitive noun phrase "Account" which makes up a CB_{NP}. It then analyses that the CB_{NP} has a sibling action (verb phrase) "registered" which does not dominate the noun phrase. As a result, S3 correctly identifies it as insensitive.

The part of speech of a word is able to affect the sensitiveness of a text. Take two sentences "*Log in*" and "*Logged in*" as an example. Both contain sensitive keyword "log" (assuming all the approaches can transform the original token to its lemma form correctly). Supor and UIPicker can correctly identify "Log in". Whyper and AutoCog fail because they are not able to identify single actions. However all the related approaches fail in identifying "Logged in". Even though it contains sensitive keyword, it is a hint text showing the user has already logged in. They fail because the part of speech of tokens is not considered in the related approaches. S3 can correctly identify such texts.

Sentiment also affects the sensitiveness of a text. For instance, the sentence "*Your Password*" describes an input box receiving user's password but "*Invalid password*" is a hint text showing the user has typed a wrong password even though it contains the sensitive keyword "password". Such negative text reduces its sensitiveness. All the related approaches do not consider the sentiment of a text, so all fail. S3 correctly identify such text.

FP/FN Analysis

In this section, we analyze the causes of false positives and false negatives under threshold 0.7. Here we select representative examples to discuss the causes.

First we present why S3 incorrectly identifies a text as some category of sensitive data, which produces false positives.

- **Inaccurate underlying NLP infrastructure**. One major source of false positives is the incorrect syntactic parsing of texts by the underlying NLP infrastructure. Take the text "Send Email" as an instance. It is not labeled as sensitive data by our volunteers. However, the underlying Stanford Parser is not able to correctly parse its syntax in original form. It annotates the whole text as a noun phrase. S3 then analyzes its topic and finally classifies it as category "Profile" with maximum probability 85.3% as it has three neighbors "email", "e-mail" and "email address" while zero neighbors in other categories. However, it correctly parses the syntax in lowercase form: an action "send" followed by a noun phrase "email". The noun phrase "email" is classified as category "Profile" but the dominant action "send" has no neighbors within threshold in category "Profile". Therefore, S3 classifies the text as insensitive in the lowercase form. Due to the classification priority, S3 chooses the category with the higher probability among original and lowercase form, such text is eventually identified as category "Profile". We observe that a majority of false positives result from incorrect syntactic parsing. Such cases can be addressed with the advancement in underlying NLP infrastructure.
- **Inaccurate threshold control**. Take the text "*Product ID:*" as an example, S3 successfully identifies it as a noun phrase "Product ID"/"product id" (in original and lowercase form respectively) with a following colon. This text matches the pattern that only contains a noun phrase. In topic analysis, S3 classifies

Table 4 Comparison of S3 and related work

Sentence	Supor	UIPicker	Whyper	AutoCog	S3
Enter your password	✓	✓	✓	✓	✓
Facebook will not save your password					✓
Register Account	✓	✓	✓	✓	✓
Account registered			✓		✓
Log in	✓	✓			✓
Logged in					✓
Your Password	✓	✓	✓	✓	✓
Invalid password					✓

✓: The sentence can be correctly identified

the text using 7 concept spaces. It finds 2 neighbors "id" and "user id" within the threshold with maximum probability 80.1% in category "Profile" while zero neighbors in other categories. Then the text is incorrectly identified as category "Profile", but actually it is a normal text describing a product. Such false positives result from inaccurate threshold control and can be addressed by increasing the threshold.

Next we present why S3 incorrectly identifies a text as insensitive, which produces false negatives.

- **Inaccurate underlying NLP infrastructure**. Consider the text "*Zip Code*". It is labeled as category "Profile" by volunteers, but S3 identifies it as insensitive. S3 correctly parses the text as a noun phrase in the original form. However, both the token "Zip" and "Code" are identified as names of entities and then removed. In the lowercase form, the text is incorrectly parsed as an action "zip" followed by a noun phrase "code". Although the noun phrase "code" has neighbors in category "Profile", the dominant action "zip" has zero neighbors in any categories so the text is identified as insensitive. In this instance, the syntax is parsed correctly in the original form but the underlying named entity analysis is incorrect. Such instances can be addressed with the improvement of underlying named entity parser. In "*Change Passcode*" of the original form, S3 incorrectly parses its syntax as a noun phrase and in the lowercase form it parses its syntax as an action "change" followed by an adjective "passcode". It is easy to know the correct syntax should be an action "change" followed by a noun phrase "passcode". Such instances can be addressed with the advancement of the underlying NLP infrastructure.
- **Incomplete knowledge base**. For instance, the word "logout" is not found in the top frequent words of our base corpus English Wikipedia 2014 and Gigaword version 5. It causes S3 to incorrectly identify the text "*logout*" as insensitive. Such issues can be addressed by collecting more words from the knowledge base corpus.
- **Incomplete concept space**. There are a few false negatives caused by the incomplete concept space. For example, the text "*Use street address*" is parsed correctly as an action "use" followed by a noun phrase "street address". However, the dominant action "use" has zero neighbors of the concept spaces of actions in any categories. Therefore S3 incorrectly identifies it as insensitive. This case can be addressed by extending concept spaces or decreasing the threshold.
- ***CB* priority**. A small source of false negatives results from the priority issue in processing *CB*. For instance, the sentence "*Allow Ad to create a calendar event?*" describes a "Setting" manner, but S3 identifies it as insensitive. S3 correctly parses its syntax as an action "Allow" followed by a noun phrase "Ad" and an action "create" followed by a noun phrase "a calendar event". S3 first identifies the sensitive candidate noun phrase "a calendar event" because it has three neighbors in category "Calendar" while noun phrase "Ad" has no neighbors in any categories. Then S3 identifies the noun phrase "a calendar event" is dominated by an action "create". This action also has neighbors in concept space of actions in category "Calendar". Then the noun phrase "a calendar event" and its dominant action "create" make up a CB_{ANP}. Since the noun phrase "Ad" is insensitive but its dominant action "Allow" has neighbors in category "Setting", the action "Allow" makes up a CB_A. Because the CB_{ANP} has higher priority than CB_A, S3 identifies the sensitiveness of the text based on the CB_{ANP} "create a calendar event". However the syntax check fails because the CB_{ANP} is dominated by another action "Allow" (the ancestor node is a verb phrase). Such issue can be addressed by processing the *CB* sequentially until it reaches a sensitive category rather than only processing the top priority *CB*.
- ***CB* definition**. Rare false negatives result from the issue in defining *CB* in cases that the noun phrase and its dominant action belonging to different categories. Take the text "*Search by location*" as an instance. It describes a "Search" manner, but S3 identifies it as insensitive. S3 correctly parses its syntax as an action "Search" followed by a noun phrase "location". The noun phrase "location" has neighbors in category "Profile", but its dominant action "Search" has no neighbors in category "Profile". Therefore the noun phrase and its dominant action cannot form a CB_{ANP}, and is thus classified as insensitive. Such issue can be addressed by forming two separate *CB*s for the noun phrase and its dominant action.

Performance

We evaluate the performance of S3 by measuring the average time of identifying a text and its memory usage. We use S3 to identify top 20,000 out of 1,741,143 texts and measure the average time for each text. The experiment is performed on a Dell PowerEdge R730 server with 20 cores (40 threads) of Intel(R) Xeon(R) CPU E5-2660 v3 @ 2.60GHz and 64 GB memory. The operating system is 64 bit Ubuntu 14.04.1 with Linux 3.19.0 kernel. The total time of processing 20,000 texts is 8682.4 s. The average time for each text is 0.43 s. Memory usage is 1,502 MB including the base corpus.

Discussion

The techniques of S3 has broader applications beyond sensitive data detection in web and mobile applications. It can also be applied to protect users' data privacy from unwitting leakage in the context of social network sharing. For example, when one shares her article, email, chat history, or even a photo to a friend, she may want her personal data (e.g., home address, school name, medical records) safely protected from leakage. However, careful manual check before each sharing is both inefficient and incomplete. Moreover, the user may even have no explicit definition of her privacy data. She can simply provide S3 with some conceptual descriptions that she cares about, and S3 is able to automatically scan the texts/images (leveraging image recognition if necessary) before she shares, and warns her if any potential privacy leakage is detected. We leave this as future work.

The potential advancement of S3 could be to automate the process of providing the conceptual descriptions by users. For instance, the user can select several articles and images that she thinks contain sensitive data but are hard to explicitly elaborate. For example, some paragraphs may contain her profile information, or some part of the image has her car number. S3 can hopefully learn the implicit specifications of sensitive data from such sample articles and images using machine learning techniques, and thus can automatically identify similar sensitive data in unseen articles and images without involving users to manually provide conceptual descriptions. We leave a further investigation into it as future work.

Related Work

To the best of our knowledge, S3 is the first systematical tool to automatically identify sensitive data including input data and output data from descriptive texts in mobile applications. Our approach utilizes NLP and learning based methods to analyze descriptive texts. Related research efforts using NLP and/or learning based methods to analyze texts/documents mainly are: 1) Sensitive input data identification in Android applications (Nan et al. 2015; Huang et al. 2015); 2) Detecting mismatches between Android UIs and program behaviors (Avdiienko et al. 2017); 3) Description/Review-to-Behavior fidelity analysis in Android applications (Pandita et al. 2013; Qu et al. 2014; Yu et al. 2016; Kong et al. 2015); and 4) Automatic discovery of Indicators of Compromise (IOC) (Liao et al. 2016).

Sensitive input data identification in Android applications. Supor (Huang et al. 2015) analyzes the descriptive texts of input boxes to analyze their sensitiveness. It first locates all the input boxes of a UI and then searches for their descriptive text. It uses keyword based searching to analyze such texts, and thus could cause many FP and FN because no semantic and syntactic information are considered. Moreover, the process of generating keywords needs much manual effort and also lacks flexibility of involving new sensitive data categories. UIPicker (Nan et al. 2015) utilizes SVM (Support Vector Machine) to learn the descriptive texts. The features are a set of sensitive keywords. The accuracy tends to increase as the size of training set increases. However it is also the limitation of this approach, because it causes much manual effort to prepare a well-labeled training set. The features are also limited by the size of sensitive keywords, so that it cannot handle unknown words. Compared with such approaches, S3 considers complete semantic and syntactic information to give accurate sensitiveness of a descriptive text. Besides, S3 does not require much manual effort to prepare massive keywords or training set.

Detecting mismatches between Android UIs and program behaviors. BackStage (Avdiienko et al. 2017) checks the advertised functionality of Android UI elements (e.g., buttons) against their implemented functionality to detect such mismatches. To get the advised functionality, it analyzes the descriptive texts of these elements. It collects all the verbs and nouns from their application dataset and then clusters them into 250 classes. It gets the advertised functionality by testing the membership of a target UI element among the 250 clusters. The approach of BackStage is similar to the topic analysis of S3, but it does not consider syntactic information. Therefore, BackStage is able to get only the approximate meaning of a descriptive text, and is thus not applicable to identifying sensitiveness.

Description/Review-to-Behavior fidelity analysis in Android applications. Approaches are proposed to identify the real permissions an Android application needs from its descriptions (Description-to-Behavior fidelity) or users' reviews (Review-to-Behavior fidelity). AutoCog (Qu et al. 2014), Whyper (Pandita et al. 2013), and TAPVerifier (Yu et al. 2016) analyze Description-to-Behavior fidelity. AutoCog uses a learning based method to generate a dataset of noun phrases with corresponding verb phrases and possessives (called *np-counterpart* in the paper) if any. It performs an np-counterpart based searching on descriptions to identify the real permissions. Actually it is an extension of keyword based searching, and does not consider complete syntactic information as well. Whyper first extracts related noun phrases and actions from API documents. Then it checks if the noun phrase is dominated by the action in a description. It considers syntactic information, but it is not complete, and no other semantic information (e.g., POS, sentiment) is considered. TAPVerifier first collects verbs in different actions and then defines semantic patterns of descriptions. However, only verbs are not sufficient to analyze the sensitiveness

of a text in our problem domain. AUTOREB (Kong et al. 2015) analyses Review-to-Behavior fidelity. The approach of AUTOREG is similar to UIPicker. It also uses a machine learning method and the features of the classifier are keywords as well. The difference is that AUTOREB utilizes the "relevance feedback" technique (Xu and Croft 1996) to add relevant words to the keyword list. Syntactic information is not considered in AUTOREG either.

Automatic discovery of Indicators of Compromise (IOC). IOC is an artifact observed on a network or in an operating system that with high confidence indicates a computer intrusion. It can be converted into a machine-readable OpenIOC format for automatically analysis. iACE (Liao et al. 2016) is proposed to discover IOC data in online pages (e.g., blogs, forums) and creates IOC in OpenIOC format. It first identifies IOC sentences in a document by searching IOC tokens and context terms. It identifies IOC tokens with regrexes and uses keyword based searching to identify context terms. Then it checks the relation between IOC tokens and context terms by graph mining. Finally, it creates IOC if the relation passes the check. Though iACE considers relatively complete syntactic information, regrex matching of identifying IOC tokens and keyword based searching of identifying context terms could cause many FPs and FNs.

In our problem domain, it is intractable to standardize sensitive data because different users may care about different sensitive data. Therefore it requires an approach that allows users to specify sensitive data on demand. This is the key technique of S3. It proposes the notion of *concept space* to represent a category of sensitive data. None of the related approaches have such flexibility to define a category on demand. S3 also performs syntax, POS and sentiment analysis to further enhance the identification result while prior work does not consider the complete semantic meaning of sensitive data.

Conclusion

Users' privacy preference is often implicit, and the definition of sensitive data is subjective and flexible. In this paper, we propose S3, a novel approach to identify sensitive data from implicit user requirements. S3 takes semantic, syntactic and lexical analysis into account to understand the semantic meaning of sensitive data and then decides its sensitiveness. We propose the notion *concept space*, which is constructed by initial readable concepts provided by users. S3 is able to learn users' preferences from the concept space, and automatically identify related sensitive data. We evaluate S3 using more than 18,000 applications from top 500 free applications in 34 categories of Google Play. We classify sensitive data into seven categories. S3 achieves an average precision of 89.2%, and average recall 95.8% in identifying sensitive data.

Acknowledgments
We thank for the reviewers for their valuable comments.

Funding
This research is supported by the National Research Foundation, Prime Ministers Office, Singapore under its National Cybersecurity R&D Programme (Grant No. NRF2015NCR-NCR002-001).

Availability of data and material
The dataset supporting the conclusions of this article is available in the Dropbox repository https://www.dropbox.com/s/00hjztyw2zw35i3/s3.zip?dl=0.

Authors' contributions
The authors have contributed jointly to the manuscript. Both authors have read and approved the final manuscript.

Competing interests
We confirm that we have read SpringerOpen's guidance on competing interests. None of the authors have any competing interests in the manuscript.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Received: 25 May 2018 Accepted: 14 August 2018
Published online: 29 September 2018

References

Amazon shopping application. https://goo.gl/aHNvpy

Avdiienko V, Kuznetsov K, Rommelfanger I, Rau A, Gorla A, Zeller A (2017) Detecting behavior anomalies in graphical user interfaces. In: 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C), Buenos Aires. pp 201–203. https://doi.org/10.1109/ICSE-C.2017.130

Baccianella S, Esuli A, Sebastiani F (2010) Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In: Chair NCC, Choukri K, Maegaard B, Mariani J, Odijk J, Piperidis S, Rosner M, Tapias D (eds). Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10). European Language Resources Association (ELRA)

Budianto E, Jia Y, Dong X, Saxena P, Liang Z (2014) You can't be me: Enabling trusted paths and user sub-origins in web browsers. In: Stavrou A, Bos H, Portokalidis G (eds). Research in Attacks, Intrusions and Defenses. Springer International Publishing, Cham. pp 150–171

Bursztein E, Soman C, Boneh D, Mitchell JC (2012) Sessionjuggler: Secure web login from an untrusted terminal using session hijacking. In: Proceedings of the 21st International Conference on World Wide Web. WWW '12, ACM, New York. pp 321–330. http://doi.acm.org/10.1145/2187836.2187880

Cunningham P, Delany SJ (2007) K-nearest neighbour classifiers. Mult Classifier Syst 34:1–17

Enck W, Gilbert P, Chun BG, Cox LP, Jung J, McDaniel P, Sheth AN (2010) Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In: Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation. OSDI'10, USENIX Association, Berkeley. pp 393–407. http://dl.acm.org/citation.cfm?id=1924943.1924971

Finkel JR, Grenager T, Manning C (2005) Incorporating non-local information into information extraction systems by gibbs sampling. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics. ACL '05, Association for Computational Linguistics. pp 363–370. https://doi.org/10.3115/1219840.1219885

Gabrilovich E, Markovitch S (2007) Computing semantic relatedness using wikipedia-based explicit semantic analysis. In: Proceedings of the 20th International Joint Conference on Artifical Intelligence. IJCAI'07, Morgan Kaufmann Publishers Inc. pp 1606–1611. https://dl.acm.org/citation.cfm?id=1625275.1625535

Huang J, Li Z, Xiao X, Wu Z, Lu K, Zhang X, Jiang G (2015) SUPOR: Precise and scalable sensitive user input detection for android apps. In: 24th USENIX Security Symposium (USENIX Security 15). USENIX Association.

pp 977–992. https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/huang
Jurafsky D, Martin JH (2000) Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. (1st ed.). Prentice Hall PTR
Klein D, Manning CD (2002) Fast exact inference with a factored model for natural language parsing. In: Proceedings of the 15th International Conference on Neural Information Processing Systems. NIPS'02, MIT Press, Cambridge. pp 3–10. http://dl.acm.org/citation.cfm?id=2968618.2968619
Kong D, Cen L, Jin H (2015) Autoreb: Automatically understanding the review-to-behavior fidelity in android applications. In: Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security. CCS '15, ACM, New York. pp 530–541. https://doi.acm.org/10.1145/2810103.2813689
LDC: English gigaword fifth edition. https://catalog.ldc.upenn.edu/LDC2011T07
Li X, Hu H, Bai G, Jia Y, Liang Z, Saxena P (2014) Droidvault: A trusted data vault for android devices. In: 2014 19th International Conference on Engineering of Complex Computer Systems. pp 29–38. https://doi.org/10.1109/ICECCS.2014.13
Liao X, Yuan K, Wang X, Li Z, Xing L, Beyah R (2016) Acing the ioc game: Toward automatic discovery and analysis of open-source cyber threat intelligence. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. CCS '16, ACM, New York. pp 755–766. http://doi.acm.org/10.1145/2976749.2978315
Lu K, Li Z, Kemerlis VP, Wu Z, Lu L, Zheng C, Qian Z, Lee W, Jiang G (2015) Checking more and alerting less: Detecting privacy leakages via enhanced data-flow analysis and peer voting. In: Proceedings of the Network and Distributed System Security Symposium (NDSS)
Mannan M, van Oorschot PC (2007) Using a personal device to strengthen password authentication from an untrusted computer. In: Financial Cryptography and Data Security. Springer. pp 88–103
Manning C, Surdeanu M, Bauer J, Finkel J, Bethard S, McClosky D (2014) The stanford corenlp natural language processing toolkit. In: Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations. Association for Computational Linguistics. pp 55–60. https://doi.org/10.3115/v1/P14-5010, http://www.aclweb.org/anthology/P14-5010
Mikolov T, Sutskever I, Chen K, Corrado G, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2. NIPS'13, Curran Associates Inc. pp 3111–3119. http://dl.acm.org/citation.cfm?id=2999792.2999959
Nan Y, Yang M, Yang Z, Zhou S, Gu G, Wang X (2015) Uipicker: User-input privacy identification in mobile applications. In: 24th USENIX Security Symposium (USENIX Security 15). USENIX Association. pp 993–1008. https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/nan
Olson DL, Delen D (2008) Advanced data mining techniques. Springer Science & Business Media
Oprea A, Balfanz D, Durfee G, Smetters DK (2004) Securing a remote terminal application with a mobile trusted device. In: 20th Annual Computer Security Applications Conference. pp 438–447. https://doi.org/10.1109/CSAC.2004.33
Pandita R, Xiao X, Yang W, Enck W, Xie T (2013) WHYPER: Towards automating risk assessment of mobile applications. In: Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13). USENIX. pp 527–542. https://www.usenix.org/conference/usenixsecurity13/technical-sessions/presentation/pandita
Pennington J, Socher R, Manning C (2014) Glove: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics. pp 1532–1543. https://doi.org/10.3115/v1/D14-1162, http://www.aclweb.org/anthology/D14-1162
Qu Z, Rastogi V, Zhang X, Chen Y, Zhu T, Chen Z (2014) Autocog: Measuring the description-to-permission fidelity in android applications. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. CCS '14, ACM, New York. pp 1354–1365. http://doi.acm.org/10.1145/2660267.2660287
Rastogi V, Chen Y, Enck W (2013) Appsplayground: Automatic security analysis of smartphone applications. In: Proceedings of the Third ACM Conference on Data and Application Security and Privacy. CODASPY '13, ACM, New York. pp 209–220. http://doi.acm.org/10.1145/2435349.2435379
Roalter L, Kranz M, Diewald S, Möller A, Synnes K (2013) The smartphone as mobile authorization proxy. In: Proceedings of the 14th International Conference on Computer Aided Systems Theory (EUROCAST). Springer. pp 306–307
Sharp R, Madhavapeddy A, Want R, Pering T (2008) Enhancing web browsing security on public terminals using mobile composition. In: Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services. MobiSys '08, ACM, New York. pp 94–105. http://doi.acm.org/10.1145/1378600.1378612
Socher R, Perelygin A, Wu J, Chuang J, Manning CD, Ng A, Potts C (2013) Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics. pp 1631–1642. http://www.aclweb.org/anthology/D13-1170
Steinbach M, Karypis G, Kumar V, et al (2000) A comparison of document clustering techniques. In: KDD Workshop on Text Mining. ACM, Boston Vol. 400. pp 525–526
Toutanova K, Klein D, Manning CD, Singer Y (2003) Feature-rich part-of-speech tagging with a cyclic dependency network. In: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1. 0NAACL '03, Association for Computational Linguistics. pp 173–180. https://doi.org/10.3115/1073445.1073478
USA Today: Driver's license creditcardnumbers:Theequifaxhackiswayworsethanconsumersknew. https://www.usatoday.com/story/tech/2018/02/10/equifax-hack-put-more-info-risk-than-consumers-knew/326260002/
Wikipedia. https://wikipedia.org
Xu J, Croft WB (1996) Query expansion using local and global document analysis. In: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '96, ACM, New York. pp 4–11. http://doi.acm.org/10.1145/243199.243202
Yahoo! data breaches. https://en.wikipedia.org/wiki/Yahoo!_data_breaches
Yu L, Luo X, Qian C, Wang S (2016) Revisiting the description-to-behavior fidelity in android applications. In: 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER) Vol. 1. pp 415–426. https://doi.org/10.1109/SANER.2016.67
Zhou Y, Jiang X (2013) Detecting passive content leaks and pollution in android applications. In: Proceedings of the 20th Network and Distributed System Security Symposium (NDSS)
Zhou Y, Evans D (2011) Protecting private web content from embedded scripts. In: Atluri V, Diaz C (eds). Computer Security – ESORICS 2011. Springer. pp 60–79

Submit your manuscript to a SpringerOpen journal and benefit from:

- ► Convenient online submission
- ► Rigorous peer review
- ► Open access: articles freely available online
- ► High visibility within the field
- ► Retaining the copyright to your article

Submit your next manuscript at ► springeropen.com