INTRODUCTION

Checking whether a program does what it claims to do is a longstanding problem for developers. Whenever we install a new app, we run the risk of the app being “malware”—that is, to act against the interests of its users.

An app that sends a text message to a premium number to raise money is suspicious? Maybe, but on Android, this is a legitimate payment method for unlocking game features. An app that tracks your current position is malicious? Not if it is a navigation app, a trail tracker, or a map application. An application that takes all of your contacts and sends them to some server is malicious? This is what WhatsApp does upon initialization, one of the world’s most popular mobile messaging applications. The question thus is not whether the behavior of an app matches a specific pattern or not; it is whether the program behaves as advertised.

Our software will attempt to check implemented app behavior against advertised app behavior. Our domain is Android apps, so chosen because of its market share and history of attacks, frauds and threat to user.

PROBLEM STATEMENT

Checking of App behavior against App description so that the user has complete knowledge beforehand whether the application is a malware for the user or not.

Our main objective is not whether the behavior of an app matches a specific pattern but whether the program behaves as advertised. The key idea is to associate descriptions and API usage to detect anomalies.

The intuition behind our project is simple: applications that are similar, in terms of their descriptions, should also behave similarly. e.g; “The ‘weather’ application which access location, accesses the messaging API, which is unusual for this category.”

Our software start with collection method for Android apps. After initial processing, it will identify topics of app descriptions, and then clusters the apps based on common topics. ***Our project is to identify outliers based on app advertisement and its description so as to prevent user beforehand from malicious apps.***

PROPOSED SOLUTION

COLLECTING APPLICATIONS

Our approach is based on detecting anomalies from “normal”, hopefully benign applications. As a base for such “normal” behavior, we collected a set of applications from the Google Play Store, the central resource for Android apps.

Fetching Description of Apps

Our software uses the method of web crawling to fetch description of the apps available at Google Play Store and also removes non-text items such as numerals, HTML tags, links and email addresses.

Preprocessing Description

Before subjecting our descriptions to topic analysis, we applied standard techniques of natural language processing (NLP) for filtering and stemming.

Multilanguage Filtering

App descriptions in the Google Play Store contain paragraphs in multiple languages—for instance, the main description is in English, while at the end of the description developers add a short sentence in different languages to briefly describe the application. To be able to cluster similar descriptions, we had to choose one single language, and because of its predominance we chose English.

Filtering

Stopwords are common words that generally do not contribute to the meaning of a sentence, at least for the purposes of information retrieval and natural language processing. We're going to create a set of all English stopwords, then use it to filter stopwords from a sentence.

Our software will remove stopwords(common words) such as ‘the’ , ‘is’, ‘at’ , ‘which’, ‘on’, ‘and’ etc.

Stemming

Stemming algorithm attempt to automatically remove suffixes (and in some cases prefixes) in order to find the “root word” or stem of a given word. Stemming improves the results of NLP, since it reduces the number of words.

LDA(Latent Dirichlet Allocation)

To identify sets of topics for the apps under analysis, our software will resort to topic modeling using Latent Dirichlet Allocation (LDA). LDA relies on statistical models to discover the topics that occur in a collection of unlabeled text. A “topic” consists of a cluster of words that frequently occur together. By analyzing a set of app descriptions on navigation and travels, for instance, LDA would group words such as “map”, “traffic”, “route”, and “position” into one cluster, and “city”, “attraction”, “tour”, and “visit” into another cluster. Applications whose description is mainly about navigation would thus be assigned to the first topic, since most of the words occurring in the description belong to the first cluster.

Clustering Apps

Topic modeling assigns an application description to each topic with a certain probability. In other words, each application is characterized by a vector of affinity values (probabilities) for each topic. However, what we want is to identify groups of applications with similar descriptions.

Table 3 shows the list of clusters that were identified. Each of these 32 clusters contains apps whose descriptions contain similar topics, listed under "Most Important Topics". The percentages reported in the last column represent the weight of specific topics within each cluster.

Identifying Outliers

Firstly, our software will download APK files and for each android application our software will extract all APIs in location.

Using all API calls as features would induce overfitting in later stages. Therefore, we select a subset of APIs, namely the sensitive APIs that are governed by an Android permission setting. These APIs access sensitive information (such as the user's picture library, the camera, or the microphone) or perform sensitive tasks (altering system settings, sending messages, etc.) So our software will identify all sensitive APIs corresponding to each APK.

Now that we have all API features for all apps, the next step is to identify outliers—that is, those applications whose API usage would be abnormal within their respective topic cluster.

BENEFITS

The intuition behind our project is simple: applications that are similar, in terms of their descriptions, should also behave similarly. e.g; “The ‘weather’ application which access location, accesses the messaging API, which is unusual for this category.” So, it would be beneficial for the user to know beforehand to know about the behavior of the app user wants to use.

It would help user to know about malware or fraud apps that are made specifically for the purpose of gaining secured information from the device.

REFERENCES

- StackOverflow.com
- AskUbuntu.com
- Digitalhistoryhacks.com
- Play.google.com
- Docs.python.org