Part 1 - The Issue

Introduction

Android phones make up about 74% of the whole smart phone market share [1]. With the abundance of Android phones on the market, malicious software, i.e. malware, can cause damages to the users. This is why malware dectection is needed to improve Android phones' security.

A technique to analyze whether an application is malware or not makes use of the application's Application Programming Interface (API) calls and their relationships with each other. These API call relationships are structured as a heterogeneous information network (HIN) [2].

The Data

The data that is used for analysis are each application's smali code. Each Android application is saved as a .apk file, which is similar to Window's .exe file or MacOS's .dmg file. Within the apk file, there exists a .dex Dalivik executable file, which is not human readable. In order to turn the dex formatted file into a readable format, APKTool is used to unzip the apk and decompile the Dalivik executable [2].

The data that will be used are: 1. Benign Android Application from apkpure.com that I will collect and decompile to Smali Code. 2. Malware samples provided as Smali Code.

Why is the data is appropriate to address the problem? - The resulting smali code that is generated using APKTool is the data that is being used to analysis. The data is valid because smali code is the "intermediate but interpreted code between Java and DalvikVM." This means that through the smali code we will be able to observe what an application is doing. From these observations, we will be able to create relationships and detect anomalies.

What are the potential shortcomings of the data for addressing the problem? - There could be app type imbalance. Apps may be games or productivity tools and they could have different pattern sequences. For example, we could have games that do simple things such as flappy bird compared to a complex game that utilizes a lot of computing power such as Asphalt, a racing game. - Bot created apps can be simple and in large numbers. This can create an issue because it can bias the whole training dataset with similar code patterns, and render the analysis useless. - How the apps are chosen can also be tricky. Do we want to randomly sample apps from a list? Do we want apps that has a similar representation of the apps that most people download? Do we want to sample apps by size, rating, or number of downloads?

What data have been used to address this problem in the past?[2] - **Dynamic analysis** – DroidDolphin: recorded activity features – Crowdroid: extracted API system calls – CopperDroid: extracted operating system interations and inter/intra-proc - **Static analysis** – DroidMat: API calls, permissions, and intent messages – DroidMiner: API calls

How data is analysed After extracting out the API calls, we make three relationship graphs. 1. APIs that co-exists in the same code block 2. APIs with the same package name 3. APIs that have the same invoke method Each API call is saved as a node, and each relationship is saved as an edge. From these reltionships, we can create a matrix. The values will be our features, and we can train out model using the matrix.

Part 2 - Data Ingestion

Where does the data come from? The data that we will be using is smali code from various Android apps. The apps' APK are downloaded from apkpure.com, and the smali code is generated using APKTool to disassemble the APK file. Then the all the smali files' directory are saved using python.

- Use two libraries, requests and BeautifulSoup, in Python to get access to the apkpure sitemap.
- Scrape all the links for an app.
- For the first stage of this project's development, randomly shuffle apps.
- Iterate until *size* apps have been converted into smali code – For each link, download the app and use APKTool to convert the app into smali code.

size from config.json file

Is this legal? The purpose of this project is to analyze whether an app is malicious or not. This purpose tries to help people protect their devices. I believe that this is an ethical and meaningful mission. In addition, this mission is similar to that of APKTool's: "It is NOT intended for piracy and other non-legal uses. It could be used for localizing, adding some features or support for custom platforms, analyzing applications and much more."

Are there any privacy issues?" There doesn't seem to have any privacy issues because we are mainly just reverse engineering the apps. The apks themselves do not contain personal information. Additionally, API calls are avaliable online, so our data will not intrude on privacy.

Storage considerations and directory layout - Downloaded xml.gz files are saved in ./xml - Downloaded apk are saved in ./apk - Downloaded smali files are saved in ./data

Configuration file so far { "size": number of apps in training set, "type": type of apps in training set }

Reference [1] https://gs.statcounter.com/os-market-share/mobile/worldwide [2] hindroid