Part 1

The problem investigated in this setting is to detect malware (malicious software) from benign applications based on applying innovatively new methods studying the smali code of the application. Nowadays, Android is dominating the smart phone market as a open source and customizable opertaing system, attracting attackers to disseminate malware, posing a serious threat to users and the smartphone market. And historically, major defense against malware attack is mobile security products, such as Norton and Lookout, which relies on signature-based method to recognize threats. Such traditional methods can be easily bypassed by techniques such as code obfuscation and repackaging, rendering them vulnerable in front of more sophiscated malwares.

Here, we implemented an innovative method based on the structured heterogeneous information network. This method analyzes different relationships between API calls within the codes and create higher-level semantics for better malware detection. Thus the data needed is smali code, a human readable representation of binary Dalvik bytecode of each Android App. To make a complete dataset, we would need both benign Android Applications and malware samples. We will collect benign Andriod App codes from apkpure.com and decompile the apk file to smali code by methods introduced in part 2. Smali codes of malware samples will be provided by class professor. Since the malware samples retrieved is from the professor, there might be limitations of sample size. Also, there might be biases in the malware set such as they are easily accessible and no more hazardous since they are provided to us.

Part 2

The data source for the benign malware Andriod Apps is from apkpure.com, which allows all user agent to scrape data based upon its robots.txt file. Also, apkpure provides https://apkpure.com/sitemap.xml for us to easily access the addresses of all kinds of Andriod Apps from the xml file. Apk files scraped will be converted to Smali code by apktool, "whose terms and conditions" will be strictly enforced. And the Smali code will be stored on personal disk with education purpose for privacy issues. Detailed data injestion process will be described below.

First of all, we will retrieve all valid xml files that contains linkages to the download page of the app. And every xml file has linkages for a certain category of Apps. Then, to ensure that our sample mimics the real-world situation mostly, we use random sample to retrieve certain number of xml files, maintaining the natural distrubtion of App categories. Then, within the xml file, we sample one link out to download the App. One possible drawback of this sampling method is that malwares might massively concentrated in an underrepresented category. Then, random sampling will make this model poor at distinguishing malwares from benign Apps in this category. Thus, another sampling method that should be tried out in the future is to enforce a certain number of samples from each category. However, this way, the natural distrubtion of each cateogry will be destroyed.

After getting the apk files, we will save them to the apk_file (decided by configuration file) directory as AppName.apk. And then, we will use apktool to convert apk files to smali code and save them to smali_file (decided by configuration file) directory as AppName folder. Since the only interest of this model is the smali code, all others will be deleted for space saving.

Data ingestion process is wrapped up as piplines with codes and configurations separated. Thus, this pipline is highly applicable towards other webpages for apk file downloads with small modifications. Thus, the data souce will be more abundent if such websites are found.